UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Aplicación Android para la visualización de contenidos educativos desarrollados con JCLIC

Miguel Nistal García
Tutor: Luis Fernando Lago Fernández

Junio 2015

Resumen

En los últimos tiempos las tecnologías de la información (TIC) se han abierto hueco en la vida cotidiana de las personas. Nos encontramos en la era de la información, en la cual casi todos los electrodomésticos, vehículos o dispositivos móviles están interconectados y permiten el acceso a la información desde cualquier lugar.

La intención de este trabajo es utilizar el avance de la tecnología para reforzar el trabajo del profesor desarrollado en el aula mediante herramientas de autor potentes y versátiles, con las cuales se crean actividades educativas. La herramienta de autor elegida es JClic, una aplicación de software libre para el desarrollo, realización y evaluación de actividades educativas multimedia, tales como rompecabezas, asociaciones, ejercicios de texto, palabras cruzadas, etc. Esta aplicación está muy extendida entre la comunidad educativa y es ampliamente usada para reforzar el contenido estudiado en la clase y acercar las nuevas tecnologías a los alumnos. Los datos de un proyecto desarrollado con JClic se almacenan en formato XML, lo cual facilita su reutilización en otras aplicaciones. El objetivo de este proyecto es conseguir desarrollar una aplicación Android que permita utilizar proyectos desarrollados con JClic en dispositivos móviles que utilicen el sistema operativo Android, consiguiendo una nueva plataforma para la realización de las actividades y aprovechando las ventajas que tienen los dispositivos móviles. Inicialmente se pretende que la aplicación desarrollada permita que funcionen las actividades tipo asociación, memoria y sopa de letras y en un futuro poder ampliar el número de actividades hasta conseguir una integración completa con el programa de autor.

Con la realización de este trabajo se espera aportar un nuevo soporte para la enseñanza que facilite la labor del personal docente, creando una aplicación completamente funcional y que sea puesta en práctica en un entorno real. El contenido de este resumen se estudiará y ampliará de manera más detallada en los siguientes apartados del proyecto.

Palabras clave

JClic, BridgeClic, Android, Comunidad educativa, XML, TIC, Docente, Sistema operativo.

Abstract

In recent times the information technologies (ICT) have opened hole in the everyday lives of people. We are in the information age, in which almost all appliances, vehicles and mobile devices are interconnected and provide access to information from anywhere.

The intention of this work is to use technological advances to enhance the work of the teacher developed in the classroom through by powerful and versatile author tools, with which educational activities are created. The tool chosen JClic author is a free software application for development, implementation and evaluation of educational activities, such as puzzles, associations, text exercises, crosswords, etc. This application is widespread among the educational community and is widely used to enhance the content studied in class and bring new technologies to students. Data from a project developed with JClic are stored in XML format, making it easy to reuse in other applications. The objective of this project is to develop an Android application that allows using projects developed with JClic in mobile devices using the Android operating system, getting a new platform for the implementation of activities and taking advantage with mobile devices. Initially it was intended that developed application allow such activities as the association, memory and word search and in the future expand the number of activities to achieve full integration with the program author.

With the completion of this work is expected to provide new support for education to facilitate the work of teachers, creating a fully functional application and is implemented in a real environment. The content of this summary will be studied and expanded in more detail in the following sections of the project.

Keywords

JClic, BridgeClic, Android, Educational community, XML, ICT, Teachers, Operating System.

Índice de contenido

Índice de Figuras

Índice de tablas

1. Introducción

1.1 Motivación

Actualmente la industria informática y electrónica está en constante expansión. Podemos encontrar dispositivos electrónicos casi en cualquier nuevo electrodoméstico o mecanismo que compremos. Especialmente está teniendo un gran auge la industria de los dispositivos móviles o Smartphone. Estos terminales cuentan con una gran aceptación entre el público y cada día el número de dispositivos vendidos aumenta. Es tal el crecimiento del sector, que ya se venden más Smartphone o Tablet que ordenadores personales, lo que produce que el desarrollo de software se empiece a centrar en estos dispositivos [1].

Este creciente interés por los terminales inteligentes también se nota en el sector educativo, que hasta la fecha se centraba más en ordenadores (que se encontraban en las aulas de informática), portátiles y pizarras interactivas. Con los nuevos dispositivos móviles se puede interactuar de manera más sencilla e intuitiva, dando más libertad al alumno para aprender, y potenciar a los niños a adaptarse a las nuevas tecnologías. Además debido a la gran cantidad de aplicaciones o *Apps* que se pueden encontrar en los proveedores, la cantidad de contenido al que se puede acceder es bastante mayor que con los antiguos dispositivos. Por este motivo la comunidad educativa cada vez está más interesada en el desarrollo de aplicaciones que sirvan para enseñar y reforzar los contenidos aprendidos en el aula de una manera interactiva que favorece la atención de los alumnos.

En la actualidad existen diversas aplicaciones educativas, cuyo uso está extendido en las aulas. Normalmente estas aplicaciones o webs requieren de una conexión a internet permanente, ya que muchas de ellas son páginas web de juegos y cursos, lo que en mi opinión dificulta la implantación en las aulas y la autonomía de los alumnos a la hora de manejar los equipos debido a la gran cantidad de contenido no apto para determinadas edades que se encuentra en la red.

Por todos estos motivos es necesario el desarrollo de herramientas educativas capaces de operar en el máximo número de dispositivos, para que el alcance sea mayor y se puedan ver beneficiados una mayor cantidad de centros educativos. Además deberían ser herramientas libre y accesibles, capaces de mostrar contenido de autor generado por los docentes, para que de esta forma el material educativo que se visualice sea adecuado de acuerdo con los métodos pedagógicos que desee usar el profesor.

1.2 Objetivo

Por todos los motivos enunciados con anterioridad se ha propuesto diseñar una aplicación Android capaz de reproducir contenido educativo realizado con herramientas de autor potentes, capaces de crear una gran cantidad de ejercicios y ampliamente usadas por la comunidad educativa.
La aplicación de autor para la que se va a desarrollar la herramienta se trata de JCLIC [2]. Esta aplicación está formada por un conjunto de aplicaciones para crear, visualizar y recoger estadísticas de actividades. El conjunto de aplicaciones está formado por:

- Author: Mediante este módulo el docente puede crear actividades como asociaciones, rompecabezas, ejercicios de texto, palabras cruzadas, ejercicios de memoria, etc. Estas actividades son mostradas en conjunto en un proyecto.
- Player: Esta parte se encarga de reproducir los proyectos generados por Jclic Author tal y como han sido creados por el equipo docente.
- Report: Además mediante esta extensión se brinda la posibilidad de recoger estadísticas de uso de las actividades, como fallos, aciertos, tiempo usado en resolver una actividad, etc.
- Applet: Un applet que permite incrustar las actividades creadas en una página web.

Con estas herramientas los profesores pueden crear material educativo para los alumnos clasificado por edades, área, idioma, etc.
Este conjunto de aplicaciones se puede usar en diversas plataformas como Windows, Linux, Solaris y MAC OS. Como se puede observar ninguna de las plataformas soportadas se trata de un soporte móvil. Aunque por otro lado al ser un proyecto de código abierto hace posible que su integración en otras plataformas sea más sencilla. El lenguaje en el que se ha desarrollado JClic es **Java** [3] y su almacenamiento de datos se realiza en fichero **XML** [4], que son de fácil lectura y transparentes para los desarrolladores.
Todo esto hace que el programa sea susceptible de ir ampliándose mediante el trabajo colaborativo, creando nuevas actividades, traduciendo la herramienta a diferentes idiomas y adaptando tanto el programa como las actividades a diferentes culturas y educadores.

A partir de aquí se decide crear una aplicación para Tablets con sistema Android capaz de reproducir un determinado tipo de actividades, que se podrán ir ampliando en un futuro, ya que el programa seguirá un desarrollo estructurado y escalable.

Como objetivo final se propone crear una aplicación Android sencilla e intuitiva capaz de reproducir las actividades creadas por el programa JClic y que pueda ser usada por niños de diferentes edades para mejorar su aprendizaje en las materias impartidas por el profesor.

1.3 Organización de la memoria

El presente documento describe las fases de diseño e implementación de una aplicación para terminales móviles. Por esto en los siguientes puntos se tratara cada paso del desarrollo de este proyecto software de manera más específica y detallada.

En el apartado 2 se estudiara el estado en el que se encuentra el desarrollo de aplicaciones para dispositivos móviles, enunciando la aportación de la tecnología a la educación, las aplicaciones existentes en el contexto del trabajo y las tecnologías usadas para la realización de este proyecto.

El capítulo 3 detallará la fase de análisis de las características de la aplicación, desde el análisis de requisitos hasta la representación de las posibles maquetas para la validación por parte del cliente.

En la siguiente sección, analizaremos como estructuraremos la aplicación, explicando la manera en la que se almacenarán y extraerán los datos, la modularización de las clases, la arquitectura y la representación (vistas) final que tendrá la aplicación.

El apartado 5 mostrará las fases de la implementación de la aplicación, detallando los módulos de los que consta y explicando los algoritmos usados para que esta sea fiable y eficiente.

La fase final del desarrollo de la aplicación se detallará en el capítulo 6, enunciando las pruebas que se realizarán para garantizar el correcto funcionamiento de la misma y mostrando y analizando los resultados obtenidos, tanto a nivel de pruebas sobre el código como de las pruebas de aceptación realizadas.

Como punto final de la memoria se darán a conocer las conclusiones obtenidas con la realización de este trabajo, así como el trabajo futuro que se pueda desarrollar a partir de aquí.

2. Estado del arte

En este apartado, estado del arte, se intenta enunciar todos los trabajos, estudios o desarrollos que se han realizado durante los últimos años en el contexto en el que se desarrolla el TFG. Esto nos sirve para comprender la motivación del trabajo y los aspectos en los que este estudio puede aportar novedades. Asimismo también nos da una visión global del estado en el que se encuentra la tecnología en las aulas y de como esta se usa para favorecer el aprendizaje y la enseñanza dentro del mundo docente.

Por otro lado también se estudiará el estado en el que se encuentra la tecnología usada, haciendo especial hincapié en las plataformas usadas para la implementación de nuestra aplicación.

2.1 Las TIC en la educación

Las TIC se han convertido durante los últimos años en uno de los pilares para la innovación no solo tecnológica, sino también para el desarrollo de nuevos fármacos, avances en medicina, mejoras de los sistemas de automoción y facilitando la comunicación entre personas aparte de otros muchos campos. Uno de estos campos en los que la tecnología está empezando a introducirse es en el sistema educativo.

Al ser el uso de la tecnología cada vez más común entre los ciudadanos, es necesario que estos aprendan a manejarlas desde pequeños, comprendiendo qué son, para qué sirven y cuáles son los riesgos que su uso puede conllevar. Para esto es necesario que la educación se centre en dos aspectos: su conocimiento y su uso.
El conocimiento de las TIC es necesario en la actualidad, no se puede comprender en estos momentos cómo funciona la sociedad sin comprender qué papel juega la tecnología en ella. Es necesario que el individuo comprenda cómo se trata su información, cómo se genera, cómo se almacena y cómo se accede a ella a través de diferentes dispositivos (PC, Tablet, Smartphone, etc.). Este aspecto es muy importante, ya que con él se consigue no solo que las nuevas generaciones sean capaces de integrarse en la era de la información, sino que también mediante su uso libre y espontáneo sean capaces de innovar y descubrir nuevos usos de la tecnología.
El segundo aspecto, el uso de las TIC, es consecuencia del primero. Mediante su uso podemos conseguir aprender y enseñar cualquier materia de una manera mucho más sencilla y eficaz, practicando con juegos o actividades en edades más tempranas, o con la búsqueda de información a través de diferentes plataformas como internet o contenido multimedia en edades más adultas [5].

La integración de las TIC en la educación es uno de los proyectos en los que está mostrando más interés la UNESCO, asegurando que "*Las tecnologías de la información y la comunicación (TIC) pueden contribuir al acceso universal a la educación, la*

igualdad en la instrucción, el ejercicio de la enseñanza y el aprendizaje de calidad y el desarrollo profesional de los docentes, así como a la gestión dirección y administración más eficientes del sistema educativo."[6].

La organización de las naciones unidas para la educación, la ciencia y la cultura aboga por facilitar el acceso, la integración y la calidad de la tecnología en las aulas para potenciar el aprendizaje de los alumnos, facilitando este en lugares donde el acceso a la información es difícil de conseguir.

Conseguir la integración de la tecnología en las aulas no es tarea fácil, ya que necesita de un gran apoyo por parte del personal docente y la comunidad educativa. El primer colectivo tiene que ser capaz de aprender el funcionamiento, las ventajas y desventajas que el uso de este tipo de sistemas puede acarrear sobre los alumnos. La comunidad educativa por su parte ha de ser capaz de entender y diseñar metodologías válidas y eficaces para que el aprendizaje y enseñanza de la materia sean más sencillos e intuitivos. Todo esto ha de ir acompañado de un reciclaje continuo para así estar siempre al día de las últimas novedades y poder inculcar el nuevo uso o tecnología a los estudiantes.

A parte de los beneficios que las TIC proporcionan en la educación de los alumnos, también el aprendizaje de estas tecnologías puede servir en un futuro. Según avanza el desarrollo de la tecnología, esta está más introducida en el funcionamiento laboral de los trabajadores, por lo que comprendiendo su funcionamiento e impulsando su uso puede tener resultados muy satisfactorios a la hora de la inserción laboral de las personas potenciando su productividad y reduciendo la curva de aprendizaje.

Vamos a enumerar las ventajas que en mi opinión el uso de las TIC pueden tener en la educación:

- Medios educativos para reforzar los temas tratados en el aula.
- Mejores plataformas para facilitar la enseñanza a personas con algún tipo de discapacidad.
- Potenciar la comunicación entre alumnos, profesores y padres.
- Mayor acceso a información por parte del estudiante y los profesores.
- Aplicando las nuevas tecnologías como el Big Data conseguir información acerca de cómo trabajan los estudiantes y poder adaptar el tipo de enseñanza a cada alumno.

También existen algunas desventajas del uso de las TIC si estas no se usan con cuidado:

- Distracción, dispersión y pérdida de tiempo de los alumnos si no tienen una correcta supervisión.
- Fiabilidad de la información. La información a la que se accede puede no ser correcta si no controlamos las fuentes de las que sacamos los datos.
- Coste de los equipos necesarios.
- Puede disminuir habilidades como la escritura si se realiza un uso excesivo.

Por ultimo cabe destacar la inversión que las instituciones están realizando para facilitar el uso de la tecnología en las aulas, incorporando equipos para que los alumnos puedan empezar a utilizarlas incluso a edades muy tempranas. También podemos observar como por ejemplo la CAM (Comunidad de Madrid) va a favorecer el aprendizaje de las nuevas tecnologías incorporando asignaturas como programación de manera optativa en los centros de educación secundaria [7].

2.2 Herramientas existentes

Para poder realizar el trabajo de forma adecuada y poder sacar los mejores resultados se tiene que realizar una búsqueda e investigación de las aplicaciones o herramientas existentes, con el fin de situarnos en el contexto del trabajo que queremos mostrar y ver en qué estado se encuentra el desarrollo de la tecnología en este ámbito.

Partiendo de esto vamos a dividir el apartado en tres sub-apartados describiendo la herramienta para la que vamos a desarrollar nuestro trabajo, y las herramientas existentes que realizan una función igual o similar a lo que queremos desarrollar.

2.2.1 JCLIC

JCLIC es la aplicación de escritorio para la que se va a desarrollar este trabajo, consiguiendo que las actividades de autor que se crean con ella se puedan visualizar y jugar en una plataforma móvil.

JCLIC [2] es un pool o grupo de aplicaciones para la creación, ejecución o evaluación de actividades. El proyecto JCLIC es la evolución de Clic, un programa creado en 1992 para la creación de actividades multimedia que ha sido utilizado durante más de 10 años por profesores, educadores o pedagogos con el fin de potenciar el aprendizaje o los conocimientos de sus alumnos. Se trata de un proyecto de software libre realizado por el Departamento de Educación de la Generalitat de Cataluña bajo una licencia Publica (GPL) [8].

Mediante este proyecto se intenta facilitar la cooperación entre los educadores para aumentar la cantidad y calidad del material a reproducir y crear una herramienta más potente y sencilla para la creación y reproducción de actividades, adaptando la herramienta a los nuevos entornos de desarrollo.

Como hemos dicho con anterioridad se trata de un conjunto de aplicaciones, el cual está formado por:

- Author: Es la herramienta de autor, permite crear, modificar y publicar las actividades y proyectos.
- Player: Aplicación para realizar las actividades en el ordenador.

- Reports: Extensión que facilita la recogida y tratamiento de datos por parte del educador para poder realizar un seguimiento de la actividad del usuario.
- Applet: Se trata de un componente para incrustar las actividades en otros programas, principalmente páginas web.

Estas aplicaciones se pueden descargar desde la página web del proyecto [2], e instalarse cada una por separado dependiendo de las necesidades de cada usuario.
Mediante ellas se puede crear y jugar a distintos tipos de actividades, entre las cuales se encuentran asociaciones, memoria, puzles, sopas de letras, exploración, identificación, respuesta escrita, crucigrama, etc.

El desarrollo de JCLIC se ha realizado con el lenguaje de programación java, un lenguaje orientado a objetos y con una amplia compatibilidad con plataformas y sistemas operativos (SO). Por esto la herramienta se puede instalar y usar en Windows, Linux o MAC OS. También, al estar bajo una licencia de tipo GPL, se permite que cualquier usuario de la comunidad distribuya o modifique el código fuente del proyecto, lo que permite añadir o modificar las funcionalidades del proyecto.

Como formato de almacenamiento de datos usa el estándar XML [4], una codificación de datos en forma de árbol que hace que la información sea transparente para el desarrollador, lo que facilita su intercambio de datos con otras aplicaciones. El proyecto JCLIC pone a disposición de los usuarios el esquema XML [9] con el que se almacenaran los conjuntos de actividades.

La estructura XML de un proyecto JCLIC parte de un elemento raíz con la etiqueta <JClicProject>, que se divide en los siguientes hijos o nodos:
- <Settings>: Rama con la información relativa al proyecto como el/los autor/es, descripción, área, edades, etc.
- <Sequence>: Identifica las actividades a mostrar en el proyecto y su orden de aparición. También informa de si se puede avanzar o retroceder de actividad.
- <Activities>: Define cada una de las actividades del proyecto, indicando la clase, estilos, descripción, funcionalidad, etc.
- <MediaBag>: Almacena todos los recursos multimedia de los que dispone el proyecto como imágenes, sonidos, etc. No es necesario que un recurso sea usado por el proyecto para que esté en esta rama.

Con estos dos estándares (Java y XML) se consigue una aplicación de arquitectura abierta, que permite añadir o adaptar funcionalidades de manera colaborativa o extender la aplicación a otras plataformas.

2.2.2 DroidClic

Se trata de una aplicación que persigue objetivos similares a los que tiene este proyecto, es decir, el proyecto DroidClic intenta facilitar la reproducción de actividades desarrolladas con JClic en dispositivos móviles (Android o IOS).
La App ha sido desarrollada por un equipo de la UPC (Universidad Politécnica de Cataluña) liderado por Marc Alier. El primer desarrollo de la aplicación fue realizado por Miriam Pujol, ex alumna de la UPC.
Actualmente el proyecto está paralizado y según la información de páginas web relacionadas con el proyecto continua en fase Beta desde 2013 [10].

2.2.3 Juegos educativos para niños

Buscando por el market de Android (Google Play [11]) existen diferentes aplicaciones con juegos para la comunidad educativa, entre las que destacamos **Juegos Educativos para niños** desarrollado por **AppQuiz** [12]. Esta aplicación se encuentra entre las más descargadas y con un buen índice de valoración.
Usando la aplicación se observa que se trata de una App de juegos educativos, en la que los juegos ya vienen preinstalados en la App y solo es posible desbloquear juegos nuevos pagando.
Esta aplicación tiene numerosas desventajas con respecto a una aplicación con la que se puedan crear juegos de autor (como la que se pretende desarrollar), ya que se gana en variedad y versatilidad de diseñar juegos para perfiles específicos. En cuanto al parecido con el proyecto propuesto no se encuentran muchas coincidencias, ya que esta App es más un juego con principio y fin, mientras que la herramienta a diseñar permite la creación de juegos por parte del educador.

2.3 Tecnologías usadas

Ahora pasamos a detallar las tecnologías que usaremos para la realización de este proyecto, indicando la tecnología, el motivo de su uso y las ventajas o inconvenientes que esta pueda tener a lo largo de las fases de análisis, diseño y desarrollo del producto.

2.3.1 Plataforma Android

La plataforma para la que se va a desarrollar el proyecto se trata de Android [13]. Se trata de un sistema operativo de código abierto, que posee la mayor cuota de mercado en Estados Unidos y Europa en dispositivos móviles.

Android fue creado por la empresa Android Inc. fundada en 2003, que en 2005 sería adquirida por parte de Google. El primer Android fue lanzado en el año 2008 bajo el apodo de Android 1.0 Apple Pie, que ya incorporaba muchas de las aplicaciones actuales pero era solo un pequeño esbozo de la dimensión que adquirió Android en los años siguientes (ver figura 2.1).
Desde que se lanzó el primer teléfono con Android este ha ido actualizando sus versiones y dándole mejores características, cambiando su interfaz gráfica para adaptarse a las nuevas necesidades y mejorando el rendimiento y gestión de memoria de los terminales.

Figura 2.1: Diferencia entre las versiones 1.0 y 5.0 de Android

La última versión lanzada por Google, a fecha de la realización de este trabajo, se trata de la 5.1.1 Lollipop. Esta versión se considera un gran salto con respecto a su predecesora, ya que incorpora una nueva interfaz más fluida y dinámica, una integración casi completa con la nube y una gestión de los recursos mucho más eficaz.

Profundizando en los aspectos técnicos de la plataforma, se trata de un sistema basado en Linux, que cuenta con soporte para procesadores ARM, x86 y MIPS y está programado en C (núcleo) y Java (User Interface). Cuenta con una amplia portabilidad a diferentes tipos de terminales, ya que se puede ejecutar en teléfonos, Tablet, TV, relojes o coches. Este es uno de los motivos por el que se escoge esta plataforma.

Las aplicaciones desarrolladas para esta plataforma deben de estar escritas en lenguaje Java. Estas una vez terminadas se deben empaquetar en archivos .apk para que se puedan instalar en el sistema [14]. Android permite la instalación de paquetes desde la nube mediante su gestor de Apps Google Play. Este permite descarga los programas creados por los desarrolladores (gratuitos o de pago) e instalarlos en el dispositivo.

Además cabe destacar la cuota de mercado de la plataforma, teniendo en cuenta que cuenta con el 81,5% de teléfonos vendidos con este sistema (ver figura 2.2). Esto hace a Android el sistema más usado y por lo tanto el más conveniente a la hora de realizar una aplicación con la que llegar a la mayor cantidad de público interesado.

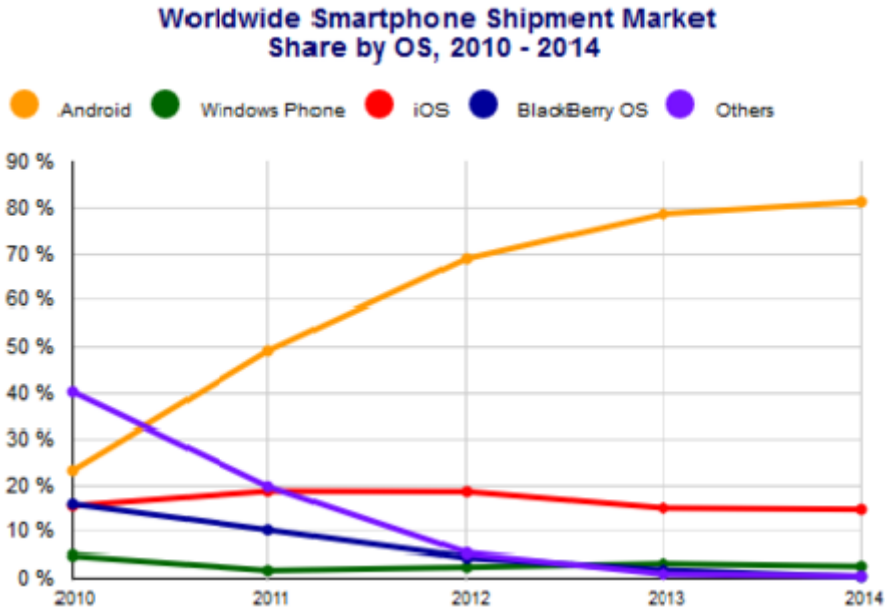

Figura 2.2: Evolución de la cuota de mercado de los diferentes SO.

2.3.2 DOM

Para la lectura de los archivos XML generados por JClic, Android dispone de dos gestores o modelos de tratamiento, SAX [15] y DOM [16]. EL primero realiza una lectura secuencial del archivo, es decir, es necesario ir realizando las acciones a medida que se lee el archivo porque de otra manera la información no estará disponible. La otra opción, DOM, realiza la lectura de una manera radicalmente opuesta, explorando primero el documento para generar después un árbol con la información obtenida. Mediante este proceso obtenemos un objeto de tipo árbol con el cual podemos ir recorriendo sus hijos o ramas para obtener la información necesaria. Para la realización de este proyecto se ha elegido la estructura DOM, ya que se adapta mejor a la lectura del fichero, pudiendo recorrer este en cualquier orden y las veces que sea preciso.

Como ventajas de este "parser" con diferencia de SAX se encuentran:

- Persistente en memoria: DOM crea una estructura que se almacena en memoria y permite recorrer esta las veces que sea necesario. Por el contrario SAX no permite volver atrás y requiere de una lectura secuencial del documento cada vez que necesitamos algo de este.
- Facilita la navegación entre nodos y el tratamiento de los datos almacenados, permitiendo visitar cualquier nodo por el nombre de su etiqueta.

La desventaja más evidente entre DOM y SAX es la eficiencia, ya que mediante DOM es necesario recorrer el archivo y después buscar cada elemento, cuando por la otra parte a la vez que se recorre el archivo se extrae la información. Por este motivo DOM no es el más adecuado para archivos XML muy grandes.

Al no ser excesivamente grandes los archivos XML generados y enfocando el diseño de la aplicación hacia un lenguaje orientado a objetos se decide usar DOM para la extracción de datos.

3. Análisis

En este apartado vamos a tratar el análisis previo al diseño y desarrollo de la aplicación. Se detallaran todos los requisitos, tanto funcionales como no funcionales, con los que deberá contar la aplicación y se incidirá un poco más en el sistema de almacenamiento de datos de la aplicación JClic que nos permitirá entender mejor su funcionamiento de cara al diseño del modelo del proyecto.

3.1 Análisis de la aplicación JClic

Como hemos descrito de manera más simple en los anteriores apartados, el programa JClic almacena en archivos XML sus proyectos. Este XML se guarda en un fichero con el nombre del proyecto y la extensión .jclic. También se guardan, bajo petición del usuario en el programa de autor, los recursos multimedia y todo junto se comprime en un Zip.

La estructura del XML se basa en 4 ramas principales, que como ya explicamos antes consisten en: ProjectSettings, Sequence, Activities y Media. Cada una guarda determinada información sobre el proyecto, el orden, el tipo de actividad y los recursos que se utilizan (ver figura 3.1).

```
<?xml version="1.0" encoding="UTF-8"?>
<JClicProject name="proyectotfg" version="0.1.3">
 <settings>
 <sequence>
 <activities>
 <mediaBag>
</JClicProject>
```

Figura 3.1: Ejemplo de XML agrupado por ramas.

Ahora vamos a pasar a detallar cada una de las ramas para poder comprender cómo se almacena la información y cómo podemos recuperar esta de la mejor manera posible (ver ejemplo XML de proyecto JClic en el Anexo 2).

3.1.1 Settings

En esta rama encontramos toda la parte referente a la descripción del proyecto creado. Aquí podemos recoger el título del proyecto, su autor o autores, su descripción, en qué lenguaje está escrito, el área a la que está dedicada, el nivel al que está orientado o incluso un icono para el proyecto. También es posible definir los sonidos que tendrá el proyecto por defecto.

Toda esta información es muy importante a la hora de configurar el proyecto y poder diferenciarlo de otros que estén dentro de la aplicación.

3.1.2 Sequence

Dentro de la rama sequence nos encontramos los nodos descriptivos de las actividades (identificador y nombre de la actividad) que nos indican el orden en el deben mostrarse en el proyecto. También ponen a disposición del usuario información de los botones que pueden ser usados en esta actividad y la condición para que se activen si es que la hay. Esto último no es de nuestro interés dentro de la aplicación como se verá en la parte de requisitos funcionales del proyecto.

3.1.3 Activities

Los hijos de este nodo Activities contienen todas las actividades del proyecto, condición que no ocurre en la rama sequence, ya que en ella solo aparecen las actividades que se van a mostrar. El principal dato que nos proporciona este nodo es el tipo de actividad descrito, pudiendo enlazar con el los demás ajustes de la actividad.

Dependiendo del tipo de actividad tiene diferentes tipos de paneles, los cuales pasamos a detallar a continuación:

- panelActivityElements: Es el panel utilizado por las actividades de asociación y memoria. Con el podemos recuperar la información de las celdas (texto, imagen, sonidos, identificador de su pareja, etc.) y de su agrupación (número de filas y columnas). También podemos ver el estilo que tendrán, entendiendo por estilo el color, tamaño o posicionamiento en la pantalla.
- textActivityElements: Este panel permite almacenar toda la información relativa a las actividades de texto o preguntas. Este tipo de panel no cuenta con celdas, sino que tiene una lista de botones (checkbox) y textos para poder resolver la pregunta. En nuestro caso no usaremos este tipo de panel debido a las limitaciones impuestas en los requisitos (ver sección 3.2). Cabe destacar que aunque estos paneles no sean utilizados se ha seguido una arquitectura en la aplicación mediante la cual se puedan introducir nuevos paneles sin alterar la estructura.
- textGridActivityElements: Esta agrupación se usa para las sopas de letras, permite posicionar cada letra en una celda dándole un formato o estilo y diferenciando su tamaño o posición. Puede contener las letras de toda la tabla o solo las de la solución. Tiene todas las posibilidades en cuanto a información de celdas, igual que en panelActivityElements.

Los únicos paneles que usaremos en el proyecto serán panelActivityElements y textGridActivityElements. El detalle de cómo se agruparan las clases se mostrará en el capítulo de Diseño del presente documento (ver sección 4).

3.1.4 MediaBag

Esta rama indica los recursos de los que dispone el proyecto, indicando como mínimo el nombre del recurso y el archivo en el que se encuentra. Si se escoge que los recursos se guarden con el proyecto la ruta del archivo contendrá solo el nombre y la extensión del archivo. En caso contrario aparecerá la ruta completa del archivo. Esta segunda opción no es muy recomendable debido a la posible diferencia de direcciones entre distintos dispositivos. En nuestra aplicación será necesario que los recursos se encuentren dentro del Zip al pasarlos al terminal móvil. Por ultimo cabe destacar que no todos los recursos que aquí aparecen tienen que mostrarse en la aplicación. Es posible que se encuentren recursos que no aparecen en ninguna actividad.

El manejo de recursos, tanto gráficos como sonoros, es una cuestión muy importante en el desarrollo de soluciones para Smartphone y Tablets, debido a que la memoria en este tipo de dispositivos no suele tener un gran tamaño y por ello debe tratarse con cuidado eligiendo que se carga en memoria, cuando y cuanto tiempo [17].

3.2 Análisis de requisitos

3.2.1 Definición del proyecto

Después de analizar la aplicación JClic en sus distintos módulos, vamos a definir que queremos realizar. La idea inicial del proyecto es diseñar una aplicación capaz de reproducir o emular distintos tipos de actividades generadas a través de la aplicación de autor JClic en dispositivos móviles que cuenten con el sistema operativo Android. A partir de esta idea se han ido añadiendo matices o requisitos adicionales. Para proponer y analizar estos nuevos requisitos se ha contado con la colaboración de un profesor del colegio **CEIP Príncipe de Asturias de Madrid,** que ha aportado una visión desde el punto de vista educativo para poder estructurar un proyecto tecnológico a un nivel educativo que sea beneficioso para la comunidad.
Esta sería la idea principal del proyecto sobre la que a continuación, con los puntos tratados en la reunión mantenida con Isaac (profesor del colegio) y Luis (Tutor del TFG) el día 4 de Noviembre de 2014, vamos a describir los requisitos establecidos para la aceptación del proyecto.

3.2.2 Requisitos funcionales

En este apartado se enumeran y explican los requisitos a nivel de funcionalidad de la aplicación. Esta funcionalidad debe de estar presente en la versión final para que pueda aceptarse el proyecto.

Actividades

R1: Asociaciones:

- Ofrecer la posibilidad de resolver ejercicios de asociaciones. Este tipo de actividad cuenta con 2 tipos de sub-actividades, asociaciones simples y complejas (ver figura 3.2 Izquierda). Se deben soportar ambos tipos.

R2: Memory

- Ofrecer la posibilidad de resolver ejercicios de memoria (ver figura 3.2 Centro).

R3: Sopas de letras

- Ofrecer la posibilidad de resolver ejercicios de sopa de letras (ver figura 3.2 Derecha).

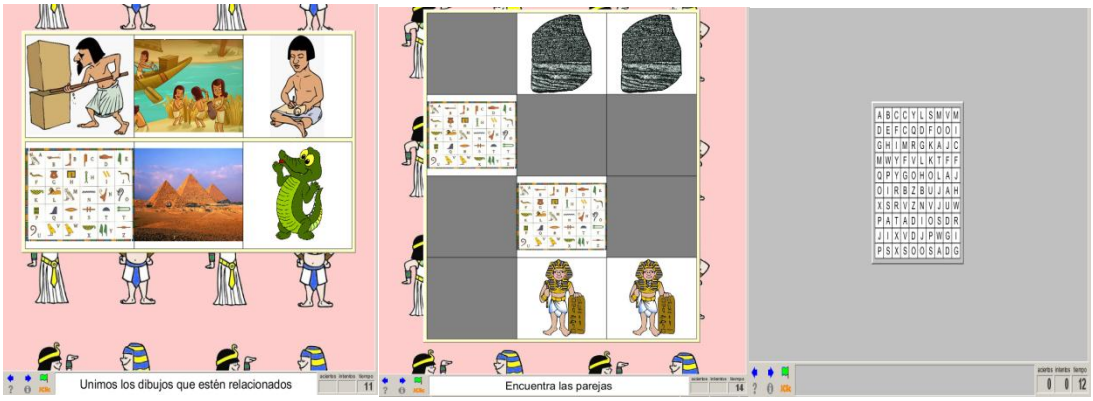

Figura 3.2 Ejemplo actividades asociación, memoria y sopa de letras.

Interfaz Actividades

R4: Intuitiva

- Debe tener botones y accesos grandes para que sean manejables para niños pequeños.
- La interfaz estará compuesta de colores planos para que tenga mejor claridad.
- Los colores de las pantallas de juego serán los especificados en el proyecto JClic, respetando en la medida de lo posible los estilos que especifique el autor.

R5: Orientación

- La orientación deberá ser siempre horizontal, para evitar distracciones con el movimiento del dispositivo.

R6: Navegación

- Se mostrarán botones de avanzar o retroceder actividad en todo momento, excepto cuando no existan actividades posteriores o anteriores.
- El botón back hará las funciones del botón salir.
- No bloquear el avance o retroceso de actividad por parte de los usuarios aunque se indique en el proyecto.

R7: Indicaciones

- Si se está cargando algún recurso deberá mostrarse un indicador de carga en la ventana.
- Mostrar aciertos, fallos y tiempo de cada actividad.
- Indicar la descripción de la actividad en un lugar visible (arriba o abajo centrado).

R8: Mensajes acústicos

- Se usarán indicaciones de voz para describir la tarea al inicio de esta.
- Reproducir sonido de acierto y fallo indicados en el proyecto nativo.
- Se dará la posibilidad de escuchar la descripción de la actividad al pulsar sobre ella.

R9: Salir

- La aplicación no debe permitir su cierre de forma libre. Se deberá mostrar un mensaje antes de que el usuario pueda salir pidiendo la confirmación del mismo.
- Se pedirá confirmación ante el evento de salir, tanto de las actividades como de la aplicación.

Interfaz Menú

R10: Pantalla inicial

- La pantalla inicial estará compuesta de una lista con los proyectos JClic existentes en la carpeta de la aplicación.
- Si los proyectos disponen de icono (imagen) para su presentación en la pantalla este debe estar visible al lado del nombre.

R11: Configuración.

- No se generará ninguna pantalla de configuración. De esta forma los usuarios más pequeños no tienen distracciones y se facilita la navegación. Todas las configuraciones del proyecto se pueden dar en la herramienta de autor.

3.2.3 Requisitos no funcionales

En este apartado se enumerarán los requisitos no funcionales de la aplicación. Estos requisitos especifican los requisitos a nivel de funcionamiento de la aplicación, no de funcionalidad.

Hardware

R12: Tamaño de pantalla

- La aplicación debe poder usarse de forma óptima en pantallas de 10 o más pulgadas.
- La aplicación está orientada a *Tablet,* por lo que se puede bloquear su instalación en móviles.

Software

R13: Versión Android

- Debe poder ejecutarse en versiones anteriores de Android (4.0 en adelante). Con esto se consigue que la aplicación esté disponible para el 90% de los dispositivos con SO Android.

Conectividad

R14: Internet

- No es necesaria la conexión a internet.

Este sería el catálogo de requisitos obtenido y sintetizado después de la reunión. Una vez acordados los requisitos realizaremos unas maquetas para fijar los requisitos y un posible aspecto de la aplicación.

3.3 Representación

En este apartado vamos a definir las maquetas que se mostrarán para la aceptación del proyecto. Por otro lado observaremos algunos casos de uso de la aplicación, para entender de una manera más grafica como deberá interactuar la aplicación con los diferentes autores.

3.3.1 Casos de uso

Para entender cómo funcionará la aplicación en los diferentes escenarios en los que se va a encontrar vamos a diseñar unos casos de uso que muestren como debería reaccionar ante las diferentes entradas o solicitudes.

Caso de uso 1: Almacenar un proyecto.
Actor primario: Docente.
Precondiciones: El profesor tiene instalada la aplicación y proyectos JClic en el ordenador.
Garantía de éxito: Se almacena el proyecto en la aplicación en un tipo de datos que esta entienda y se muestra en la lista para que el usuario lo pueda seleccionar.

Escenario principal:

1- Un profesor transfiere un proyecto JClic a la carpeta de la aplicación.
2- El usuario inicia la aplicación.
3- El sistema descomprime el archivo.
4- Los datos se copian a otra carpeta para su posterior tratamiento.
5- La aplicación informa de forma satisfactoria al usuario de que el tratamiento de datos ha finalizado.
6- El sistema muestra una lista con los proyectos existentes.

Excepciones:

3- El sistema no puede descomprimir el archivo porque este esta corrupto o tiene fallos.

E.1 El sistema detiene la ejecución e informa al usuario de que el proyecto no se puede descomprimir.
E.2 Se cancela el caso de uso.

4- Los datos no se pueden tratar debido a que el contenido del archivo no es el requerido por la aplicación.

E.1 El sistema informa al usuario de que hay proyectos no válidos y continua la ejecución.

Caso de uso 2: Visualización de un proyecto
Actor primario: Alumno.
Precondiciones: El alumno tiene instalada la aplicación y con proyectos válidos almacenados.
Garantía de éxito: El usuario puede visualizar e interactuar con las actividades de los proyectos almacenados.

Escenario principal:

1- El usuario inicia la aplicación.
2- La aplicación prepara los recursos y muestra la lista de proyectos.
3- El usuario elige un proyecto de la lista.
4- El sistema muestra la primera actividad del proyecto.
5- El usuario juega a todas las actividades del proyecto.
6- El sistema le da la opción de salir.
Se vuelve al paso 3. Se repiten los pasos 3, 4, 5 y 6 hasta que el usuario decida salir de la aplicación.

Estos serían los dos casos de uso que se podrían dar en nuestro sistema, de los cuales se puede ver una representación gráfica en la figura 3.3.

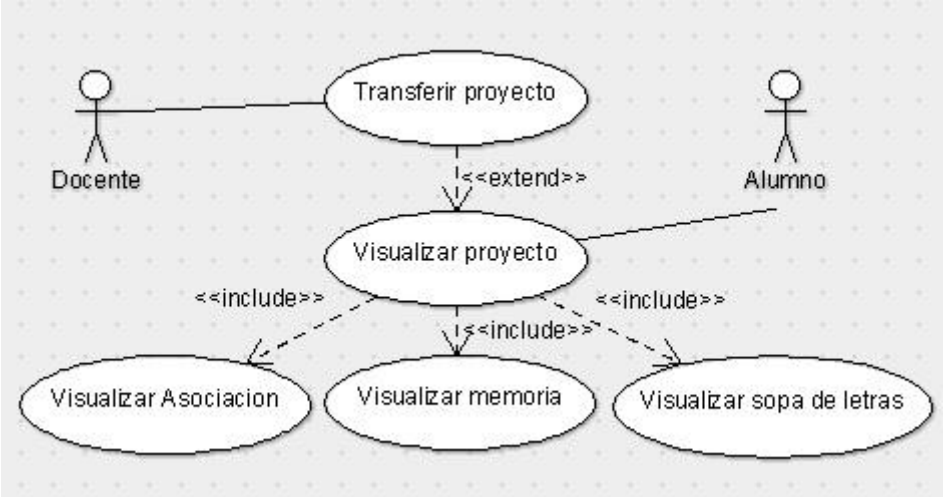

Figura 3.3: Diagrama de casos de uso de la aplicación.

La cantidad de casos de uso que puede tener la aplicación es limitada, ya que se destina a un uso muy particular en el que el profesor genera el contenido para los alumnos. Por esto básicamente contamos con dos casos, generar y transferir el contenido al sistema, y usar el sistema para visualizar las actividades propuestas por el profesor.

3.3.2 Maquetas

La siguiente fase es realizar las maquetas para que el usuario final pueda observar cómo sería un aspecto aproximado de la aplicación y entienda mejor si eso es lo que quiere o desea algún tipo de cambio.

Las maquetas mostradas al profesor son las siguientes:

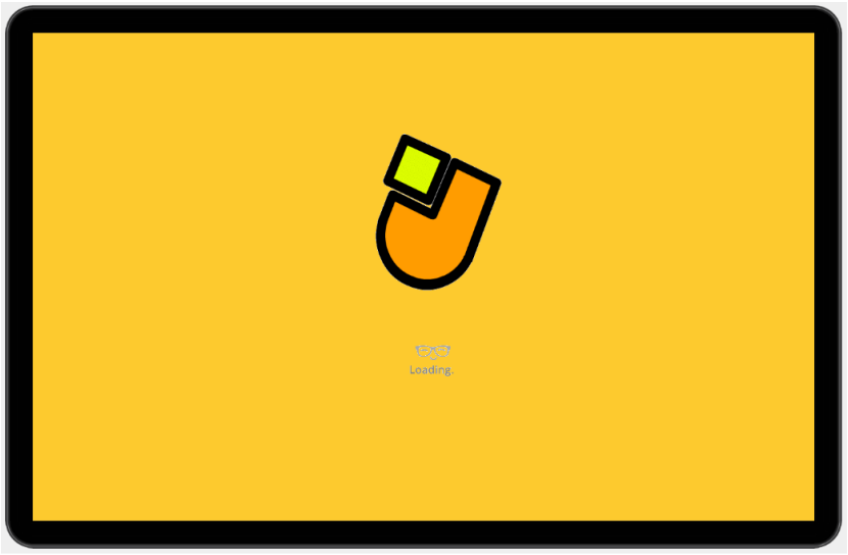

Figura 3.4: Maqueta de carga de la App.

Figura 3.5: Maqueta de selección de proyectos.

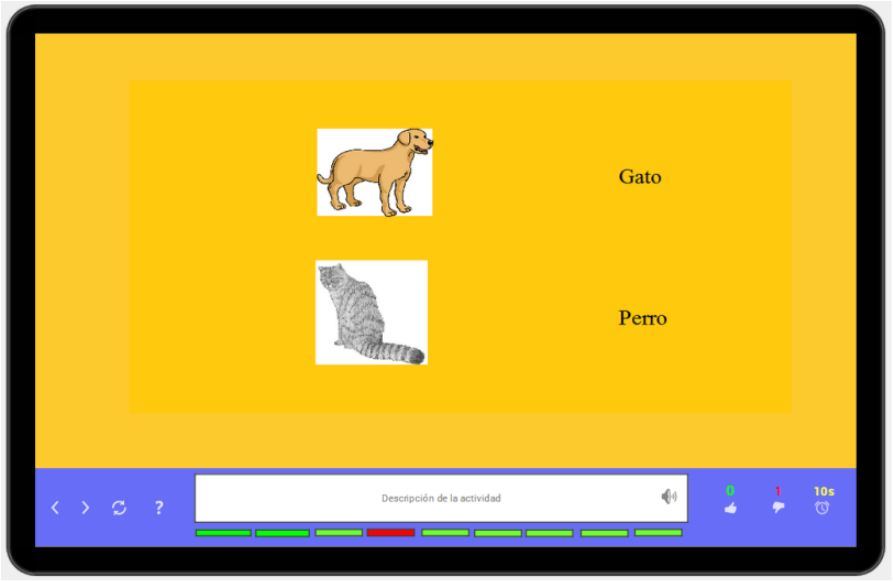

Figura 3.6: Maqueta de juego con actividad de ejemplo (Asociación).

Estas son las maquetas mostradas al usuario final. Se solicita un cambio en el tamaño de los nombres de los proyectos para hacer estos más grandes y más fácilmente accesibles a usuarios de corta edad.

Entrando un poco más en detalle con las maquetas, la primera maqueta (Figura 3.4) muestra un ejemplo de pantalla de carga, la cual deberá aparecer cuando se esté recuperando o generando algún recurso. La siguiente maqueta (Figura 3.5) visualiza la lista de proyectos disponibles para interactuar. La disposición de esta pantalla como principal es uno de los requisitos del proyecto. Por último se muestra un ejemplo (Figura 3.6) de cómo será la ventana de juego, poniendo especial hincapié en el menú inferior en el que se colocaran los diferentes botones de control, los conteos de estadísticas y la descripción del proyecto.

Mediante estas maquetas el usuario es capaz de comprobar el aspecto que podrá tener la aplicación y decidir si eso es lo que quería o desea otra cosa. En nuestro caso las maquetas se aceptaron y se sigue adelante con el proyecto.

4. Diseño

Partiendo del análisis previo vamos a diseñar una aplicación capaz de cumplir con todos los requisitos impuestos por el usuario final, tanto funcionales como no funcionales.
En este punto hay que diferenciar el qué se quiere desarrollar del cómo se va a hacer. El proceso para identificar qué queremos desarrollar y representarlo se realiza en la fase de análisis (anterior apartado) y en esta fase se explica cómo se quiere realizar el proceso para cumplir con los objetivos marcados anteriormente.
Para comenzar la fase de diseño vamos a empezar explicando el modelo a seguir para la división de funcionalidades.

4.1 MVC (Modelo-Vista-Controlador)

La arquitectura de ficheros a implementar por la aplicación será el patrón de MVC [18]. Se trata de una arquitectura de software utilizada en sistemas donde se usan interfaces de usuario. Mediante este patrón se separa el código en tres capas diferentes, acotadas por su responsabilidad dentro del sistema. Este tipo de arquitectura surge de la necesidad de crear aplicaciones más robustas orientadas a los usuarios, que potencien su mantenimiento, reutilización de código, separación de conceptos y se pueda aplicar un ciclo de vida más adecuado a los requerimientos del usuario.

Este modelo es perfecto para el tipo de diseño que necesitamos en el proyecto, ya que queremos que la aplicación sea fácil de mantener y mejorar añadiendo nuevas funcionalidades, como nuevos tipos de actividades, en un futuro.

Las ventajas de usar esta arquitectura están orientadas la mayoría a mejorar la calidad y el mantenimiento de código. Algunas de las ventajas de su uso podrían ser:

- La implementación se realiza de forma modular, lo que ayuda a dividir funcionalidades entre los diferentes fragmentos de código.
- Cada capa tiene un cometido muy claro, pudiendo centrar el desarrollo de cada capa de forma individual siempre que se haga un diseño apropiado para la solución. Para esto es necesario que el diseño sea claro y todo el equipo de desarrollo conozca la vía de comunicación entre las capas.
- La modificación de una capa no implica la modificación de todas las capas del proyecto, ya que podríamos decir que las capas están separadas entre sí.
- Para desarrollos orientados a objetos es un modelo perfecto que favorece la extensibilidad y el mantenimiento del proyecto.

Las desventajas que puede conllevar su uso se orientan más al tiempo de diseño de la aplicación, ya que requiere un diseño más exhaustivo dividiendo el proyecto en módulos muy bien definidos. Tampoco es recomendable usar este tipo de patrón en modelos no orientados a objetos, ya que su implementación resulta muy costosa y difícil en lenguajes que no implementan este paradigma.

Una vez explicadas las ventajas y desventajas por las que nos decantamos a usar este modelo para el proyecto, vamos a describir cada una de las capas en las que se divide la arquitectura:

- Modelo: Es un conjunto de clases u objetos que representan la información de círculo que el sistema quiere reflejar. Es la parte encargada de representar la información que la aplicación va a utilizar. Por ejemplo, de manera simplificada para nuestro caso, un proyecto tendrá un modelo que representará las actividades, los recursos, la configuración, etc. Normalmente el modelo cuenta con una parte de datos llamada **modelo de dominio** y otra parte que genera las interfaces con las que otra capa pueda acceder a determinada información y recibir notificaciones sobre cambios en la información del modelo. A esta segunda parte se la denomina **lógica de negocio**, la cual prepara la información del modelo para poder mostrarla a las demás capas.
- Vista: La definición más simple de esta capa es como la representación visual de los datos contenidos en el modelo. La vista se encarga de mostrar los datos del modelo y adaptarse a los cambios que este pueda notificar. No es necesario que exista una única vista para cada módulo del modelo, es decir, pueden existir varias vistas que representen la información de un modelo.
- Controlador: Contiene el código necesario para realizar las acciones solicitadas por el usuario. Sirve de puente entre el modelo y las vistas. Por ejemplo, el usuario accede a la vista y realiza una acción, el controlador la captura y solicita que se realice esa acción en el modelo. Una vez realizada notifica a la vista para que actualice sus datos. La vista puede pedir más información al modelo sin necesidad del controlador en caso de necesitarla.

En la figura 4.1 podemos ver visualmente la arquitectura a implementar.

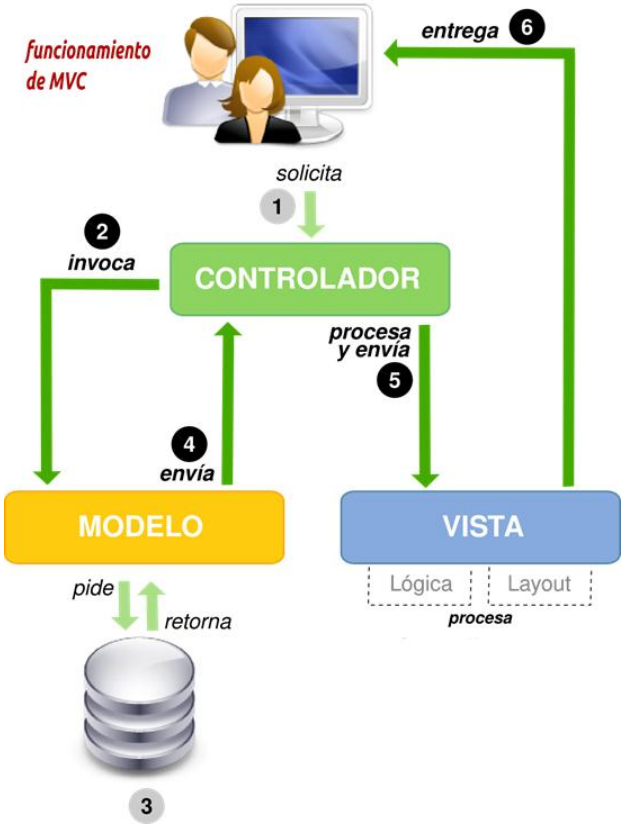

Figura 4.1: Ejemplo gráfico de MVC.

Para el diseño e implementación de este proyecto se ha decidido usar la arquitectura MVC debido a que Android estructura los ficheros de la misma forma. En el desarrollo de aplicaciones para Android se divide el código en layouts y módulos de código. Los layout se refieren a las vistas y se especifican mediante un XML, al igual que las diferentes animaciones que esta pueda tener. De diferente manera la lógica y controladores se implementan en lenguaje java, que favorece el uso de este tipo de arquitectura.

4.2 Modelo de datos

Ahora pasamos a describir los diferentes módulos de los que constará nuestro modelo, que nos ayudarán a estructurar nuestra información y a hacerla más accesible a las otras capas.

Definir los módulos o clases que pasarán a componer nuestro modelo de datos es una tarea complicada, en la que se deben tener en cuenta distintas variables para que la estructura de datos haga la aplicación eficiente, extensible y de fácil mantenimiento.
Por ello se van a separar las clases de forma parecida a cómo lo hace la aplicación nativa. Con esto conseguimos tener una adaptación más sencilla ante cualquier posible

cambio que pueda haber en la aplicación por parte de sus creadores, y favorecer la extensión de la aplicación a nuevos tipos de actividades (ver Figura 4.2).

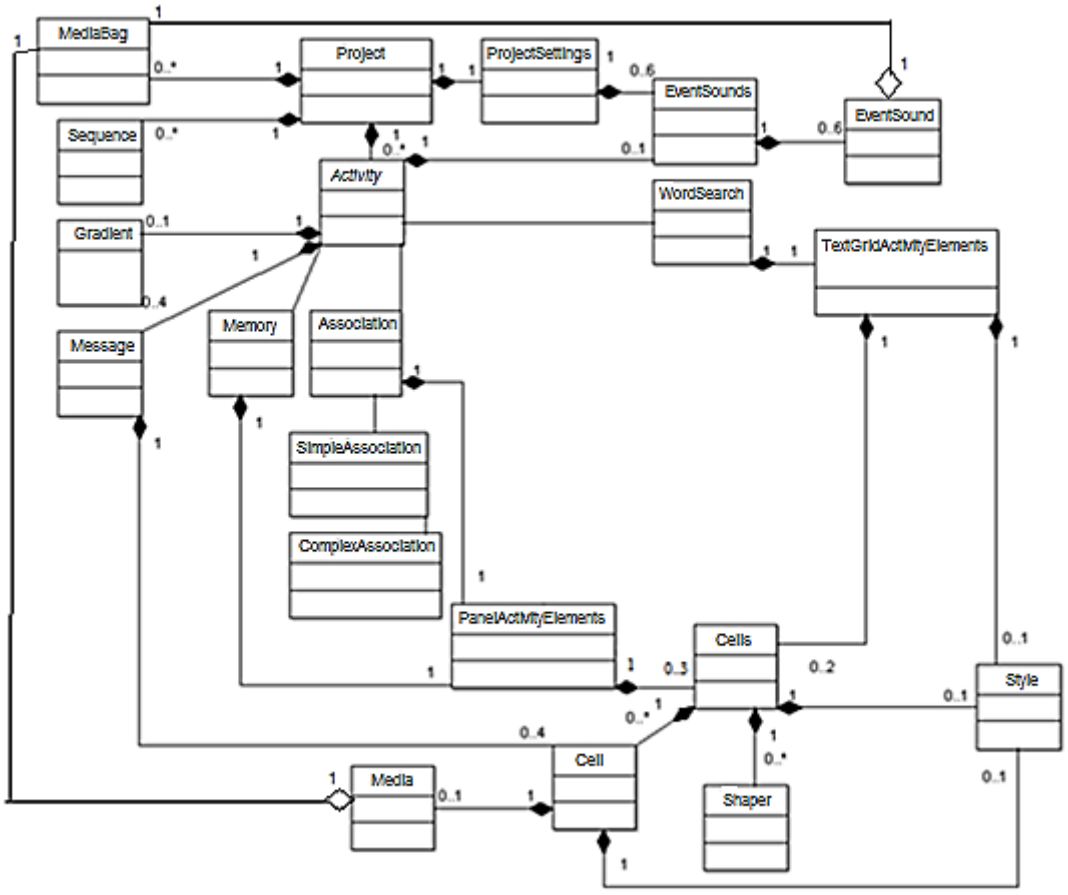

Figura 4.2: Diagrama de clases de la aplicación.

En el diagrama anterior se refleja el diagrama de clases de la aplicación[1]. En el tenemos como raíz del modelo la clase Project, que será la encargada de almacenar todas las actividades y configuraciones, tanto del proyecto como de las propias actividades.
Para entender mejor cada una de las clases vamos a pasar a detallarlas:

- Activity: Clase abstracta, contiene todos los atributos comunes de las actividades. También está compuesta por otras clases que definen la configuración de la actividad. Cada actividad puede tener sus propios sonidos, diferentes a los que tiene el proyecto.
- MediaBag: Clase que almacena un recurso del proyecto, ya sea visual o sonoro. Contiene el nombre y el archivo en el que se encuentra el recurso. Las clases Media y EventSound contienen el nombre del recurso y las propiedades con las que se utilizará.
- ProjectSettings: Guarda los atributos de la configuración del proyecto, como su nombre, nivel, área, descripción, autor, etc. También registra los recursos sonoros que tiene por defecto el proyecto.

[1] No se han podido añadir los atributos y los métodos de las clases a la figura 4.2 por motivos de espacio, ya que algunas clases como *Activity* tienen una gran cantidad de atributos.

- Sequence: Registra el nombre de cada actividad y su orden en el proyecto. Podríamos decir que se trata de una lista de actividades con el número de posicionamiento de cada actividad.

Estas serían las clases principales de las que está compuesto un proyecto. A partir de aquí la clase *Activity* pasaría a ser el centro del diagrama. De esta clase heredan sus atributos las clases *Memory, Association* (Abstracta, tiene implementaciones como SimpleAssociation y ComplexAssociation) y *WordSearch.* Las dos primeras están compuestas por un tipo de panel y la última por otro, que disponen de distintos tipos de estructuras:

- PanelActivityElements: Panel que usan las actividades *Memory* y *Association.* Este tipo de panel está compuesto por un conjunto de celdas (entre 0 y 3 conjuntos) que sirven para generar los paneles primario, secundario y el panel de ayuda.
- TextGridActivityElements: Usado por la actividad *WordSearch,* este panel cuenta con un máximo de dos conjuntos de celdas y una serie de filas que almacenan las palabras ocultas en el panel. También cuentan con otros atributos que indican los caracteres reemplazables para generar los paneles aleatorios. Por ejemplo puede aparecer el texto **"<row>**ABC*******</**row>**", donde el carácter '*' es un carácter a reemplazar aleatoriamente por la aplicación. El carácter a reemplazar viene dado por el esquema XML.

Estructurando de esta manera el modelo se pueden crear nuevas actividades sin tener que retocar el modelo, bien sea añadiendo un tipo de actividad que utilice uno de los paneles ya creados (solo existe un tipo de panel más en la aplicación nativa, *textActivityElements*) o añadiendo una actividad y su respectivo panel.

Por ultimo tenemos las celdas, que forman parte del conjunto *Cells.* Estas celdas cuentan con diferentes configuraciones para visualizarse con distintos estilos. Esas configuraciones se almacenan en las clases *Style* y *Shaper.* También pueden mostrar imágenes o reproducir sonidos, por lo que cuentan con la clase *Media* para gestionar este tipo de recursos. La clase *Media* contiene el nombre del recurso (el fichero en el que se encuentra está en la clase MediaBag) y las propiedades con las que se tiene que utilizar. La clase *EventSounds* contiene una lista con los diferentes sonidos que puede tener un proyecto o actividad, clasificados por acción (click, inicio, fallo, etc.).

4.3 Controlador

La función del controlador será realizada por las actividades Android (extienden la clase *Activity*). Estas son las clases provistas por el sistema para manejar la información que se muestra en cada pantalla, es decir, enlazan la parte lógica con la visual de cada pantalla. La actividad por defecto también asigna y captura los eventos producidos en su

vista, por lo que se convierte en una opción ideal para utilizarla como controlador.
Para este proyecto se generarán inicialmente 3 clases o actividades que harán las veces controlador. Estas clases serán:

- SplashActivity: Actividad inicial, se encarga de notificar al modelo el inicio de la aplicación para que este cargue los datos y una vez este finalice informar a la vista para que se modifique.
- ProjectListActivity: Esta clase será la encargada de mostrar los diferentes proyectos y permitir al usuario interactuar con ellos. Cuando se seleccione un proyecto avisará al modelo para que recupere la información de dicho proyecto y actualice la vista.
- GameActivity: Controlador del juego. Gestionará la transición de vistas de actividades y los eventos que estas puedan lanzar. También llevará a cabo la gestión de los recursos visuales y sonoros, indicando que recurso se muestra o reproduce en cada momento.

Con estos módulos se cubrirá la funcionalidad de la aplicación en una primera versión, pudiendo ampliarse en un futuro para añadir nuevas funciones.

4.4 Vista

Las vistas se generan o lanzan a partir del controlador (Actividad de Android). Estas vistas se definen mediante un XML que indica los elementos que contiene y su estilo, pudiendo ser retocada por el controlador tras recibir una notificación para que actualice alguno de sus valores.
La gestión de las vistas se hará mediante la clase Fragment que usa Android para pintar parte de la vista. Un Fragment es un contenedor de una vista u otros Fragments que es independiente de los demás elementos de la pantalla y puede ser reemplazado o eliminado en cualquier momento.
Cada Fragment se puede considerar una actividad dentro de la actividad principal, ya que pone a disposición del desarrollador las herramientas necesarias para controlar los eventos que se generen dentro de propio Fragment.
Por tanto para este proyecto se crearán una vista (XML) para cada actividad mencionada en el anterior apartado, es decir, una vista para SplashActivity, otra para ProjectListActivity y otra para GameActivity.

En el caso de ProjectList se incorporará un Fragment para mostrar los datos del proyecto seleccionado (ver figura 4.3). También la clase GameActivity dispondrá de Fragments. Se usará un Fragment para dibujar el cuadro de mandos del proyecto y otro para mostrar la actividad en uso (ver figura 4.4). Este último Fragment se cambiará al avanzar o retroceder de actividad favoreciendo así el uso de una sola actividad para toda la lógica de juego, dejando a cada Fragment controlar los eventos propios de cada tipo

de actividad. Mediante esto conseguimos un diseño mejor modulado que permite añadir actividades sin cambiar la lógica del juego.
Con esto damos por finalizado el diseño de la arquitectura a implementar por la aplicación, mediante la cual se intenta conseguir la máxima de flexibilidad a la hora de desarrollar y poder crear una aplicación lo más modular y extensible posible de acuerdo con la arquitectura MVC.

Figura 4.3: Fragments ProjectList.

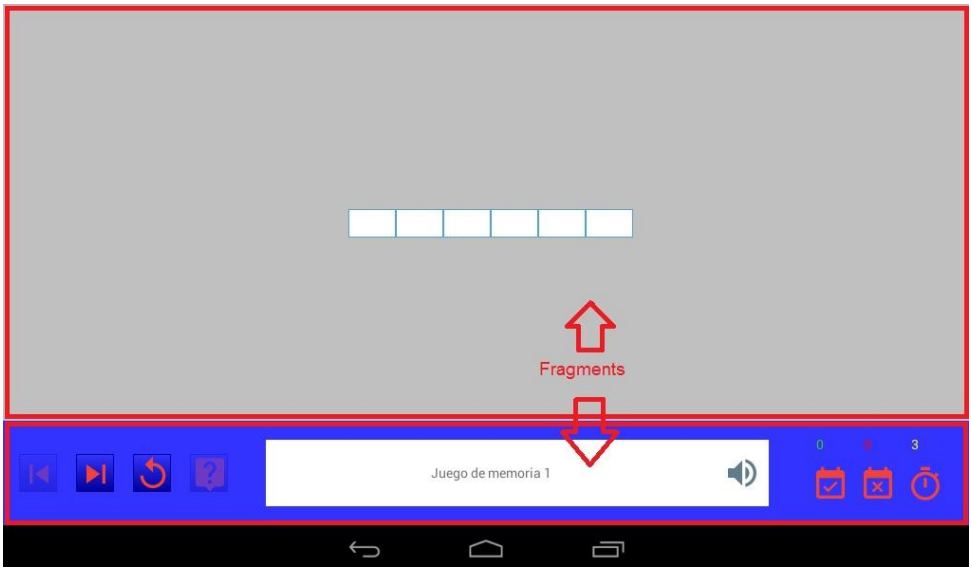

Figura 4.4: Fragments GameActivity.

5. Desarrollo

En esta sección explicaremos el desarrollo de la App, describiendo el material usado y los módulos creados siguiendo las pautas marcadas en el análisis y diseño del sistema.

5.1 Entorno de desarrollo

Google pone a disposición de los desarrolladores dos vías para crear aplicaciones para Android:

- Instalar el SDK: Esta opción permite trabajar con diferentes IDE o desde la línea de comandos, permitiendo al desarrollador configurar su propio entorno de desarrollo.
- Android Studio [19]: Es el IDE oficial de Android. Está basado en el IDE IntelliJ, añadiendo funcionalidades propias de los proyectos Android al entorno.

En nuestro caso hemos usado el IDE oficial de Android, *Android Studio*, para adaptarnos a los nuevos entornos de desarrollo propuestos por Google, que incorporan una conexión total con la nube y facilitan la distribución de las App en el mercado. También al ser un producto destinado al desarrollo para esta plataforma tiene un mejor manejo de las opciones del lenguaje y un mejor renderizado de las vistas orientado a los dispositivos en los que se va a ejecutar.

5.2 Desarrollo de módulos

Ahora pasamos a detallar el desarrollo de cada uno de los módulos, haciendo especial hincapié en las distintas funciones para poder ver como se realiza cada una de las acciones del sistema.

5.2.3 Modelo

El desarrollo del modelo se ha realizado siguiendo el diagrama de clases mostrado en el anterior apartado, dividiendo las clases para poder derivar la funcionalidad a cada uno de sus módulos de forma equitativa.

En primer lugar se han creado las clases, cada una de ellas con su conjunto de datos y funciones de acceso (ver figura 5.1).

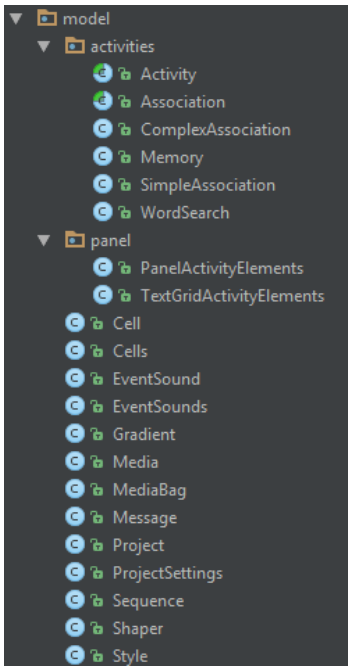

Figura 5.1: Árbol de clases del modelo.

Una vez tenemos el esquema de clases, se ha procedido a realizar la funcionalidad de cargar un proyecto desde un XML. Aprovechando el sistema escogido para el tratamiento de XML (DOM), se ha implementado en cada clase una función para obtener los datos pertenecientes a una rama del mismo tipo. El prototipo de función implementada en las clases sería el siguiente:

```
boolean readXML(Node n)  o  boolean readXML(NodeList list)
```

Recibiendo Node si el elemento a leer es una hoja o NodeList si el elemento es una rama con varios "hijos". Con esto conseguimos que cada clase se encargue de la carga de sus elementos, véase el siguiente ejemplo:

```
/**
 * Funcion para leer los valores de la clase a partir de un nodo DOM.
 * @param n Nodo de la estructura DOM.
 * @return true, si la funcion a leido los parametros correctamente. false en caso contrario.
 */
protected boolean readMediaXML(Node n){
    if(n == null){
        return false;
    }

    Element e = (Element) n;
    try{

        this.name = e.getAttribute("name");
        this.file = e.getAttribute("file");

        if(!e.getAttribute("save").equals("")){
            this.save = Boolean.parseBoolean(e.getAttribute("save"));
        }

        if(!e.getAttribute("usage").equals("")){
            this.usage = Short.parseShort(e.getAttribute("usage"));
        }

    }catch(NumberFormatException exception){
        System.out.println("Excepcion trhow in readMediaXML, class @MediaBag, Exception info: " + exception);
        return false;
    }

    return true;
}
```

Figura 5.2: Ejemplo función readXML del módulo MediaBag.

El seguimiento de la carga partiría desde la clase Project, leyendo sus variables e invocando a las clases inferiores para que lean sus valores a partir de los nodos o listas de nodos que les pase como parámetro.
La clase *Activity* propone una función abstracta para cargar los datos del XML (ver figura 5.3) que deberán implementar todas las actividades que quieran formar parte de la clase. Así conseguimos que si se añade una nueva actividad esta tenga que incorporar el lector para que sea aceptada.

```
public abstract boolean readActivityContextXML(NodeList list);
```

Figura 5.3: Función abstracta de la clase *Activity* para la carga desde XML.

La mayoría de las clases carece de mayor lógica que la de leer sus datos de un archivo plano, exceptuando la clase Project y las actividades:

- Project: Esta clase se encarga de gestionar la entrada de información para todo el proyecto, abriendo, tratando y cerrando el fichero de carga, reconociendo actividades no soportadas, calculando el estado de avance del proyecto o si una actividad tiene o no una siguiente.
- Activity: La clase Activity tiene la lógica común a todas las actividades, gestionando los mensajes, los aciertos y fallos y las interfaces necesarias para notificar al controlador o las vistas que una actividad ha finalizado en cualquiera de sus posibilidades.

La funcionalidad de cada una de las actividades recae sobre la instancia de cada una de ellas, gestionando sus propias peculiaridades. Por ejemplo, el acierto o fallo de un movimiento en la sopa de letras es diferente al de una asociación, o un posible acierto es diferente entre una asociación simple y una compleja. Cada clase presentará una interfaz para que el controlador pueda notificar los eventos y la vista recuperar la información.

5.2.3.1 Módulo WordSearch

El módulo de la sopa de letras debe tener en cuenta dos acciones, ver si un movimiento es posible y ver si este es acertado. Para ver si el movimiento es posible miramos las siguientes condiciones:

- ejexA = ejexB
- ejeyA = ejeyB
- ((ejexA - ejexB) = (ejeyA - ejeyB))
- ((ejexA - ejexB) = ((ejeyA - ejeyB) * -1))

Con las dos primeras opciones comprobamos si se marcan casillas en la misma línea, ya sea vertical u horizontal. Mediante las dos últimas se comprueba para las diagonales.
Una vez tenemos el movimiento aceptado como correcto comprobamos si la palabra entre las casillas es una de las buscadas comparando con las incógnitas almacenadas en una lista del modelo. Esta lista es leída del propio XML del proyecto. El juego acaba cuando se encuentran todas las incógnitas.

5.2.3.2 Modulo Memory

Esta clase no tiene algoritmos complicados, siendo su comprobación más sencilla que en el anterior caso. En esta ocasión la comprobación de la validez del movimiento es ver que la casilla inicial y final no son la misma y que ninguna de las dos es una de las acertadas. Una vez comprobado esto nos valemos de un hashmap para almacenar las parejas y una vez recibidas las dos celdas como parámetros se comprueba si existe coincidencia entre la clave y el valor o viceversa. Una vez acertadas todas las parejas se acaba el juego.

5.2.3.3 Modulo Association

Las asociaciones son prácticamente iguales a la hora de las comprobaciones de movimientos válidos, estas siguen los mismos principios que en la clase Memory y solo precisan de la comprobación de igualdad de celda. La diferencia entre la simple y la compleja es que en la compleja una celda no está resuelta hasta que lo están todas las que apuntan a ella, mientras que en la simple solo puede existir una relación biyectiva.
Por tanto para resolver las funciones simples necesitamos un hashmap con las parejas al igual que antes, pero para resolver la compleja nos valemos de una función que nos indica si es necesario deshabilitar la celda porque esta ya tiene todas las parejas resueltas. El juego acaba cuando todas las relaciones están resueltas.

5.3 Integración de los módulos

Para unir la funcionalidad del modelo con el entorno Android usaremos las clases del controlador, gestionando cada uno de los eventos posibles (ver figura 5.4).

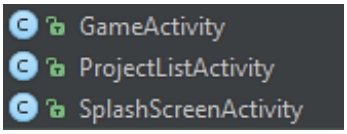

Figura 5.4: Clases del controlador.

Para comprender mejor qué hace cada uno de los módulos del controlador pasamos a detallar su implementación:

- SplashScreenActivity: Es la puerta de entrada al proyecto, estando declarada como actividad de inicio en el *Manifest* del sistema. Una vez ejecutado el proyecto esta actividad se pone en primer plano y despliega la interfaz gráfica perteneciente a la clase. Una vez inflada la vista lanza un proceso en segundo plano[2] para descomprimir los proyectos y colocarlos de una manera más manejable y a su vez cargar las variables descriptivas (nombre, descripción, área, etc.) para poder mostrarlas en la siguiente pantalla. Usamos esta pantalla

[2] Se lanza un proceso en segundo plano ya que puede tardar bastante tiempo dependiendo de la cantidad de proyectos a cargar y el tamaño de su contenido multimedia.

para realizar los procesos más costosos y así liberar a las siguientes actividades y hacer que la UI de la aplicación sea más fluida. Una vez almacenada y recuperada toda la información de los proyectos el proceso notifica a la actividad y esta da inicio a la siguiente actividad.

- ProjectListActivity: Mediante este módulo mostramos los proyectos disponibles al usuario para que los pueda seleccionar de una lista. Cuando el usuario selecciona un proyecto la clase captura el evento y actualiza la descripción del mismo. Se sobrescribe el método *onBackPressed* para pedir confirmación al usuario en caso de querer abandonar la App. La descripción contiene un botón que permite al usuario iniciar el proyecto seleccionado. También ha sido necesaria la creación de un adaptador personalizado para listas con el fin de mostrar el icono y el nombre en la lista de proyectos. Una vez el usuario seleccione un proyecto y pulse "*Empezar*" se realizara una transición a la siguiente actividad.
- GameActivity: Se trata de la actividad principal del juego. Es la encargada de gestionar todos los eventos que suceden durante el juego. En primer lugar pide al modelo la primera actividad y lanza la vista correspondiente. Cuando una actividad finaliza captura el evento y realiza la transición entre las actividades (como hemos dicho antes se reemplazarán los Fragments). En todos los posibles casos de finalización de cada actividad mostrará un mensaje informando al usuario de la acción realizada. A su vez el modelo notificará al controlador los recursos acústicos necesarios para gestionarlos de forma correcta. En este caso también es necesario modificar el método *OnBackPressed* para pedir confirmación al usuario como se solicita en los requisitos.

Con estos tres módulos podemos controlar todos los eventos que lanza la aplicación y actuar de manera correcta.

La gestión de los eventos de cada una de las distintas actividades correrá a cargo de los Fragments. Cada Fragment controlará los eventos propios de una actividad o parte de una vista de la aplicación. La estructura seguida para la creación de Fragments es la siguiente:

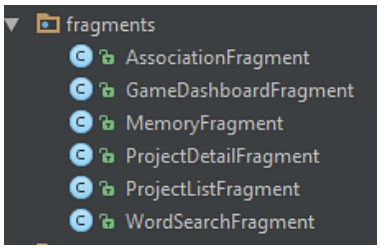

Figura 5.5: Clases Fragment.

Crearemos un Fragment para cada tipo de actividad que controle los eventos como *Onclick*, realizando las acciones adecuadas e informando al modelo y la vista. También se creará un Fragment para el cuadro de mando del juego de modo que sea adaptable y

pueda ser cambiado con facilidad en el caso de que se requiera en un futuro. Finalizando los Fragments crearemos uno para mostrar la lista de proyectos y otro para el detalle de cada proyecto, haciendo posible realizar la transición entre detalles.

Por ultimo cabe destacar las clases usadas para la carga y reproducción de audio en el dispositivo. Empezaremos por la clase *MediaPlayer*, la cual usamos para cargar un archivo de audio y reproducir su sonido. Además contiene diversas funciones, como *seekTo* o *setLooping*, que nos ayudan a reproducir el sonido de la misma forma que está diseñado con la herramienta de autor (ver figura 5.6).

```
//Comprobamos la existencia de audio en el proyecto
if(!m.getMessage().getMediaType().getFile().equals("")){
    //Reiniciamos el reproductor
    media = resetPlayer();
    //Encapsulamos la instruccion
    try {
        //Obtenemos el audio
        media.setDataSource(this.directory + m.getMessage().getMediaType().getFile());
        //Preparamos la salida
        media.prepare();
        //Avanzamos hasta donde nos indica el proyecto
        media.seekTo((m.getMessage().getMediaType().getCdFrom() == -1) ? 0 : m.getMessage().getMediaType().getCdFrom());
        //Marcamos la opcion de bucle
        media.setLooping(m.getMessage().getMediaType().isLoop());
        //Iniciamos el audio
        media.start();
    } catch (IOException e) {
        e.printStackTrace();
    }catch (IllegalArgumentException e) {
        e.printStackTrace();
    } catch (IllegalStateException e) {
        e.printStackTrace();
    }
}
```

Figura 5.6: Ejemplo de uso de la clase MediaPlayer.

Para hacer más accesible la aplicación a niños de temprana edad que no saben leer o personas con algún tipo de discapacidad se requiere que algunos textos de la aplicación puedan ser leídos por la aplicación (Catalogo de requisitos, R8). Esto lo conseguimos mediante la clase *TextToSpeech.* Esta clase nativa de Android disponible desde la API 4[3] permite reproducir en formato audio cualquier cadena de texto en un determinado idioma. La limitación reside en los idiomas instalados o preinstalados en el SO. Si el proyecto está en un idioma no soportado se pasará a reproducir en Español (Es, es) y si este tampoco se encontrara disponible en el sistema se utilizaría el lenguaje por defecto. Una vez inicializada la clase con un lenguaje, leer un texto es tan fácil como llamar a la función *speak* (ver figura 5.7).

```
reader.speak(toSpeak, TextToSpeech.QUEUE_FLUSH, null);
```

Figura 5.7: Ejemplo de uso de la clase TextToSpeech.

Con estas dos clases gestionamos todos los sonidos que se reproducen en el transcurso de una actividad o de la aplicación.

[3] Android 1.6 DONUT.

5.4 Generación de las vistas

Para finalizar con el desarrollo falta implementar las vistas de las que dispondrá la aplicación. Se crearán vistas para las actividades y Fragments (ver figura 5.8).

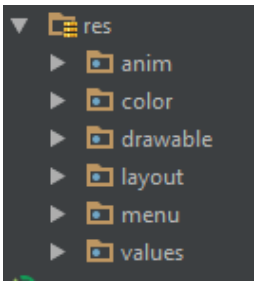

Figura 5.8: Estructura de los recursos.

Siendo la carpeta *anim* el directorio de las animaciones, *color* las definiciones de los colores, *drawable* los recursos gráficos, *layout* las vistas, *menu* los menús de las vistas (en nuestro caso no existen) y *values* las definiciones de los valores de la aplicación.

Las animaciones de las vistas se crean de la misma forma que estas, codificando un fichero XML que describa el funcionamiento de la animación. Se crearán animaciones tanto para los eventos, por ejemplo la selección de un elemento, como para las transiciones de actividades (ver figura 5.9).

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:shareInterpolator="false">
    <translate
        android:fromXDelta="0%" android:toXDelta="100%"
        android:fromYDelta="0%" android:toYDelta="0%"
        android:duration="700" />
</set>
```

Figura 5.9: Animación para mover un elemento a la derecha.

Como hemos explicado antes se generará una vista para cada actividad. Las vistas de las actividades que contienen Fragments tendrán como elemento o recurso a mostrar un tipo de Fragment, que será el que hemos creado nosotros. En caso de que el Fragment esté inicialmente vacío se usara el elemento *FrameLayout*, que permite instanciar una ventana que no tenga nada en su interior para posteriormente reemplazarla. Cuando se crea la vista notificará al Fragment asociado que cree su propia vista y este devolverá a la actividad la vista para que la incruste en el layout principal (ver figura 5.10).

```
<LinearLayout android:id="@+id/gameLayout" xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingTop="16dp"
    tools:context="es.uam.eps.tfg.bridgeclic.GameActivity"
    android:orientation="vertical"
    android:background="#FF8000">

    <FrameLayout android:id="@+id/game_container"
        android:layout_width="match_parent"
        android:layout_height="match_parent" android:layout_weight="3" />

    <fragment
        android:id="@+id/dashboard_container"
        android:name="es.uam.eps.tfg.bridgeclic.fragments.GameDashboardFragment"
        android:layout_width="match_parent"
        android:layout_height="110dp"
        tools:layout="@layout/fragment_game_dashboard" />


</LinearLayout>
```

Figura 5.10: Fichero XML de la vista para la actividad Game.

Tanto las vistas de las actividades como las de los Fragments se han implementado en distintos tamaños para poder dar soporte al mayor número de dispositivos posibles. Concretamente se han implementado para los tamaños sw600, sw720 y mayor de sw720. El sistema los reconoce directamente y asigna la vista que mejor se adapta a la pantalla del dispositivo[4].
También para facilitar la creación de recursos gráficos se ha usado la herramienta web *Android Asset Studio* que genera estos recursos en los diferentes tamaños requeridos por el sistema, de modo que sea visible en las distintas resoluciones que puedan tener los dispositivos (Ejemplos de las vistas en el manual de usuario, Anexo 1).

5.5 Configuración y permisos

Para que nuestra aplicación cumpla con los requisitos expuestos en el catálogo es necesario configurar el sistema y poner unos límites de uso. Para ello usaremos el *AndroidManifest*, que es el encargado de especificar todas las configuraciones que tendrá nuestra aplicación. Serán necesarios dos requisitos no funcionales a cumplir por la aplicación, la versión mínima del sistema y que el dispositivo permitido para ejecutar nuestra aplicación sea una Tablet (ver figura 5.11).

```
<uses-sdk android:minSdkVersion="14" android:targetSdkVersion="21"/>

<supports-screens android:smallScreens="false"
    android:normalScreens="false"
    android:largeScreens="false"
    android:xlargeScreens="true"
    android:anyDensity="true"
    android:requiresSmallestWidthDp="600"/>
```

Figura 5.11: Implementación de los requisitos.

[4] Para visualizar las pantallas finales consultar Anexo 1: Manual de usuario.

Usaremos la etiqueta <uses-sdk> para especificar la mínima versión requerida y la versión a la que está destinado el proyecto. Como versión *target* es recomendable poner la última versión del sistema para así obtener las últimas funcionalidades. Para establecer como requisito que el dispositivo utilizado sea una Tablet utilizamos la etiqueta <supports-screens>, que nos permite especificar el tamaño mínimo de pantalla que debe tener el dispositivo, así como otro tipo de parámetros. Las Tablets tienen una pantalla con un mínimo de ancho de 600dp.

Por otro lado también especificaremos los permisos que requiere nuestra aplicación para ser instalada:

- Permiso de almacenamiento: Se necesitarán permisos para leer y escribir en la memoria externa de la aplicación. En caso de no existir Android genera un directorio auxiliar emulando a la memoria externa. Este directorio es el visible al conectar el dispositivo en un ordenador.
- Permiso para bloquear la pantalla: La aplicación no permitirá que la pantalla se apague mientras se esté ejecutando.
- Permiso para matar procesos: Se usará para limpiar la memoria del sistema y que no queden procesos abiertos una vez se finalice la ejecución de la aplicación por parte del usuario.

Esto lo conseguiremos con la etiqueta <uses-permission>, indicando el tipo de permiso requerido (ver figura 5.12). Para poder instalar la aplicación el usuario debe aceptar los permisos requeridos en la configuración.

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="android.permission.KILL_BACKGROUND_PROCESSES" />
```

Figura 5.12: Permisos de la aplicación.

6. Pruebas

En este apartado se detallará el esquema de pruebas seguido para comprobar el correcto funcionamiento de los módulos, la integración con el sistema y las pruebas de aceptación por parte de los usuarios.

6.1 Pruebas unitarias

Las pruebas unitarias tratan de encontrar fallos en el código de la aplicación, buscando posibles comportamientos no deseados de las funciones que lo componen. Para realizar estas pruebas nos ayudaremos de las nuevas herramientas disponibles en el IDE Android Studio a partir de la versión 1.1. Añadiremos a las dependencias de compilación del proyecto la necesidad de importar JUNIT, que nos ayudará a realizar las pruebas.
En nuestro caso realizaremos primero las pruebas al modelo en busca de errores, comprobando el funcionamiento con valores normales, valores límite o valores erróneos. También se realizaran pruebas de caja blanca con el fin de descubrir errores en las instrucciones condicionales. En la tabla 6.1 se muestra un resumen de los resultados obtenidos tras la realización de las pruebas.

Modulo	**Resultado**
Activity	Error en la funcion *readDescriptionXML*: Se produce un truncamiento de la descripción en tamaños menores de 100 caracteres.
Association	✓
SimpleAssociation	✓
ComplexAssociation	✓
Memory	✓
WordSearch	✓
PanelActivityElements	✓
TextGridElements	Error en la función *readTextRowXML*: La carga de las filas no es correcta, se concatenan las filas en una sola cadena.
Cells	Error en la función *readCellsXML*: No existe la etiqueta 'columns' en el archivo XML.
Cell	✓
EventSound	✓
EventSounds	✓
Gradient	✓
Media	✓
MediaBag	✓
Message	✓
Project	✓

ProjectSettings	Error en la funcion *readDescriptionXML*: Se produce un truncamiento de la descripción en tamaños menores de 100 caracteres.
Sequence	✓
Shaper	✓
Style	✓

Tabla 6.1: Resultado pruebas unitarias sobre módulos del modelo.

Como se puede observar en los resultados (ver tabla 6.1) las pruebas fueron satisfactorias, ya que se localizaron errores en algunos módulos. Todos los errores encontrados han sido resueltos, excepto el error en la clase *Cells*, ya que se trata de un error en el programa de autor JClic. La especificación de la etiqueta es la siguiente:

```
<xs:attribute name="columns" type="xs:short" default="0"/>
```

Pero en los proyectos creados por la herramienta el campo recibe el nombre de *cols*. Por este motivo no se resuelve la incidencia, sino que se adapta el código al fallo del programa principal para que el funcionamiento sea el esperado.
Una vez corregidos los errores se relanza la batería de pruebas sin encontrar más incidencias.

6.2 Pruebas de integración y sistema

Mediante este tipo de pruebas tratamos de encontrar errores en las interfaces de los módulos, con el fin de garantizar un correcto funcionamiento y una buena comunicación entre las clases. También aprovecharemos la segunda fase de las pruebas para acoplar la aplicación con sus entornos hardware y software.

En primer lugar se realizarán pruebas de caja negra a los módulos del controlador, intentando simular eventos para ver la respuesta a dicho evento. No se encontraron errores en esta primera batería de pruebas, pero sí en una segunda pasada cambiando los casos erróneos. En esa pasada de observó que el controlador de juego permitía la anulación de celdas del mismo panel cuando la clase de la actividad era una asociación compleja. Una vez resuelto se relanzó la batería de pruebas y el resultado fue satisfactorio.

Para realizar las pruebas de sistema se usaron dos emuladores en el ordenador de desarrollo. Estos dispositivos virtuales prueban el funcionamiento de la aplicación en los casos límite para garantizar su correcta ejecución en el rango de dispositivos y sistemas a los que está destinado.

Las máquinas virtuales tenían las siguientes características:

AVD 1	**AVD 2**
Tipo: Tablet	Tipo: Tablet
Versión: Android 4.2.2	Versión: Android 4.0 (Mínima)
CPU: x86	CPU: ARM
Tamaño: 10’’	Tamaño: 7’’
RAM: 1GB	RAM: 512MB

Tabla 6.2: Máquinas virtuales de prueba.

Mediante la ejecución de la aplicación en estos entornos pudimos comprobar que la gestión de memoria no era la más óptima, ya que cuando se intentaban cargar imágenes demasiado grandes se producía una excepción del tipo *OutOfMemoryException*.
Para solventar este error se cambió la gestión de las imágenes en memoria, siguiendo las pautas indicadas en la documentación de Google para desarrolladores [17].

Para la gestión de imágenes se recomienda realizar primero un cálculo del tamaño final que ocupará la imagen en memoria, para comprobar si esta tiene espacio suficiente y en ese caso cargarla con su tamaño original o si no con un tamaño menor. También realizamos una carga de las imágenes por partes, cargando solo aquellas que se van a usar en ese momento y eliminándolas de memoria una vez finaliza su exposición. Por otro lado realizamos llamadas explícitas al recolector del sistema para que libere la memoria alojada en los Fragments una vez estos se eliminan.

6.3 Pruebas de validación

Para la realización de estas pruebas también se usaron dos dispositivos físicos facilitados por el tutor de este TFG (ver tabla 6.3). Para la batería de pruebas se utilizaron 15 proyectos de JClic descargados desde la página web [2]. Todos los proyectos usados son creados por diferentes docentes y personal educativo para estudiantes de diferentes edades y que engloban diferentes áreas. Podríamos decir que se trata de pruebas con fuego real, aunque controlando nosotros el juego.
Con estas pruebas conseguimos descubrir diferentes facetas del juego que no se correspondían exactamente con lo especificado en los requisitos. Se tuvo que realizar un reajuste del cálculo del tamaño que obtenía la sopa de letras con diferentes tamaños y filas o columnas. Se acordó intentar mantener el tamaño especificado en el proyecto y si este no se adaptaba a las necesidades del dispositivo que la aplicación lo reajuste al tamaño ideal. Aparte de esta incidencia y alguna que otra diferencia entre versiones del sistema las pruebas de validación se superaron satisfactoriamente.

6.4 Pruebas en entorno real

La aplicación se ha probado en un entorno real el día 28 de Mayo de 2015 en un aula del colegio CEIP Príncipe de Asturias de Madrid como parte de actividades educativas con dispositivos multimedia. Se ha contado con los mismos dispositivos que en las pruebas de validación (ver tabla 6.3). Las pruebas han sido llevadas a cabo por Isaac (profesor del colegio) y Luis (tutor de este TFG), informando de que el resultado de estas ha sido satisfactorio. Los alumnos jugaron a los distintos proyectos preparados para la actividad sin problemas, mostrando un gran interés en la aplicación.

Samsung Galaxy Tab 2	**Bq Edison 3**
Versión: 4.0	Versión: 4.4.2
Tamaño: 10.1”	Tamaño: 10.1”
RAM: 1GB	RAM: 2GB

Tabla 6.3: Dispositivos físicos utilizados en las pruebas.

Con estas pruebas se da por finalizado este apartado, no quedando exenta la aplicación de realizar más pruebas ante las modificaciones o en las tareas de mantenimiento sobre ella.

7. Conclusiones y trabajo futuro

7.1 Conclusiones

Con este trabajo se ha conseguido implementar una herramienta capaz de visualizar el contenido educativo desarrollado con el programa de autor JClic.
Con este sistema conseguimos facilitar otra herramienta más a la comunidad educativa que les permita reforzar las enseñanzas impartidas con elementos más intuitivos y sencillos de usar que además despiertan el interés de los estudiantes. La aplicación no solo visualiza las actividades desarrolladas en la herramienta de autor, sino que introduce nuevos diseños y formas de jugar para favorecer si inclusión en las aulas.
La herramienta ha sido creada desde un principio por y para la comunidad educativa, diseñando e implementando cada una de sus funcionalidades de acuerdo a la opinión de los docentes involucrados en el proyecto y realizando cambios o mejoras de acuerdo a las indicaciones facilitadas. También cabe destacar que mediante este sistema se consigue que estos métodos sean más accesibles a los estudiantes, reduciendo los costes de implantación al ser dispositivos más baratos que un ordenador tradicional, facilitando su almacenamiento en las aulas al tener un tamaño más reducido y mejorando su portabilidad en lugares fuera del recinto educativo.

Por último, a nivel personal, la realización de este trabajo me ha permitido comprender mejor el desarrollo de aplicaciones basadas en el usuario, aprender de una manera más autodidacta la implementación de programas para entornos móviles basados en Android y entender la labor y los métodos del personal docente para conseguir enseñar cada uno de los temas a sus alumnos.

7.2 Trabajo futuro

Uno de los pilares fundamentales en el desarrollo de esta aplicación es dotar de un grado de extensibilidad muy alto a dicha aplicación. Para ello se han seguido diferentes patrones que favorecen esta labor. El trabajo posterior a la realización de este proyecto consistiría en seguir incluyendo distintos tipos de actividades a la aplicación con el fin de enriquecer cada vez más los proyectos ejecutados en nuestro sistema.
Para poder ampliar el rango de usuarios que tienen acceso a la aplicación, sería necesario almacenarla en la nube. De esta manera los profesores de distintos colegios podrían ayudarse de esta herramienta para reforzar su labor docente.
Por ultimo cabe la posibilidad de abrir el rango de dispositivos en los que la App puede operar. Se podría adaptar para la versión móvil, retocando las vistas, o migrar la aplicación a otros sistemas operativos, como es el caso de IOS, aumentando así la cuota de mercado.

El enfoque de los desarrollos posteriores a este trabajo tiene muchas vías y solo es necesario establecer un orden de prioridades para saber qué camino tomar.

Glosario

App: Aplicación para dispositivos móviles
IOS: Apple Operating System
TFG: Trabajo Fin de Grado
AVD: Android Virtual Device
CPU: Central Processing Unit
RAM: Random Access Memory
IDE: Integrated Development Environment
XML: eXtensible Markup Language
OS: Operating System
API: Application Programming Interface
UI: User Interface
DOM: Document Object Model
MVC: Model View Controller
SAX: Simple API for XML
ARM: Advanced RISC Machine
MIPS: Microprocessor without Interlocked Pipeline Stages
GPL: General Public License
TIC: Tecnologías de la información y la comunicación
UNESCO: United Nations Educational, Scientific and Cultural Organization

Referencias

[1] “El imparable crecimiento de los dispositivos móviles”, *http://www.cookingideas.es/el-imparable-crecimiento-de-los-dispositivos-moviles-20100223.html*

[2] JClic, XTEC, *http://clic.xtec.cat/es/jclic/*

[3] Java, *www.java.com/*

[4] “Extensible Markup Language (XML)”, W3C, *http://www.w3.org/XML/*

[5] “Las TICS en los procesos de Enseñanza y Aprendizaje”, *http://educatics.blogspot.com.es/*

[6] “Las tecnologías de la información y la comunicación (TIC) en la educación”, UNESCO, *http://www.unesco.org/new/es/unesco/themes/icts/*

[7] Comunidad de Madrid, *http://www.madrid.org*

[8] “GNU GENERAL PUBLIC LICENSE”, GNU, *http://www.gnu.org/copyleft/gpl.html*

[9] Esquema XML JClic, *http://clic.xtec.cat/doc/jclic.xsd*

[10] DroidClic, *www.code.google.com/p/droidclic/*

[11] Google Play, *https://play.google.com/*

[12] Juegos educativos para niños, *http://www.robotifun.com/*

[13] Plataforma Android, *www.android.com*

[14] Publishing Android Apps, *http://developer.android.com/tools/publishing/publishing_overview.html*

[15] Simple API for XML, *http://www.saxproject.org/*

[16] Document Object Model, *http://www.w3.org/DOM/*

[17] Displaying Bitmaps Efficiently, *http://developer.android.com/training/displaying-bitmaps/index.html*

[18] Model-View-Controller, *http://en.wikipedia.org/wiki/Model-view-controller*

[19] Android Studio, *https://developer.android.com/sdk/*

Anexo

Anexo 1: Manual de usuario BridgeClic

Instalación

La instalación de la aplicación sigue las mismas pautas que cualquier otra aplicación para Android. Solo es necesario ejecutar la APK de la App y esta se instalará en el sistema. Si no se dispone de la APK almacenada en el dispositivo es necesario copiar esta en dicho dispositivo y ejecutarla. Es importante seleccionar la opción de permitir fuentes desconocidas para evitar posibles errores en la instalación. Ajustes > Seguridad > Fuentes desconocidas. Una vez instalada la aplicación tendremos un icono con el nombre **BridgeClic**, que corresponde con el nombre de la aplicación.

Copia de proyectos al dispositivo

Para poder copiar proyectos al dispositivo y que este te los reconozca es necesario copiarlos en una carpeta específica generada por la aplicación dentro del dispositivo.
La creación de esta carpeta se hará de forma automática en el primer uso de la aplicación.

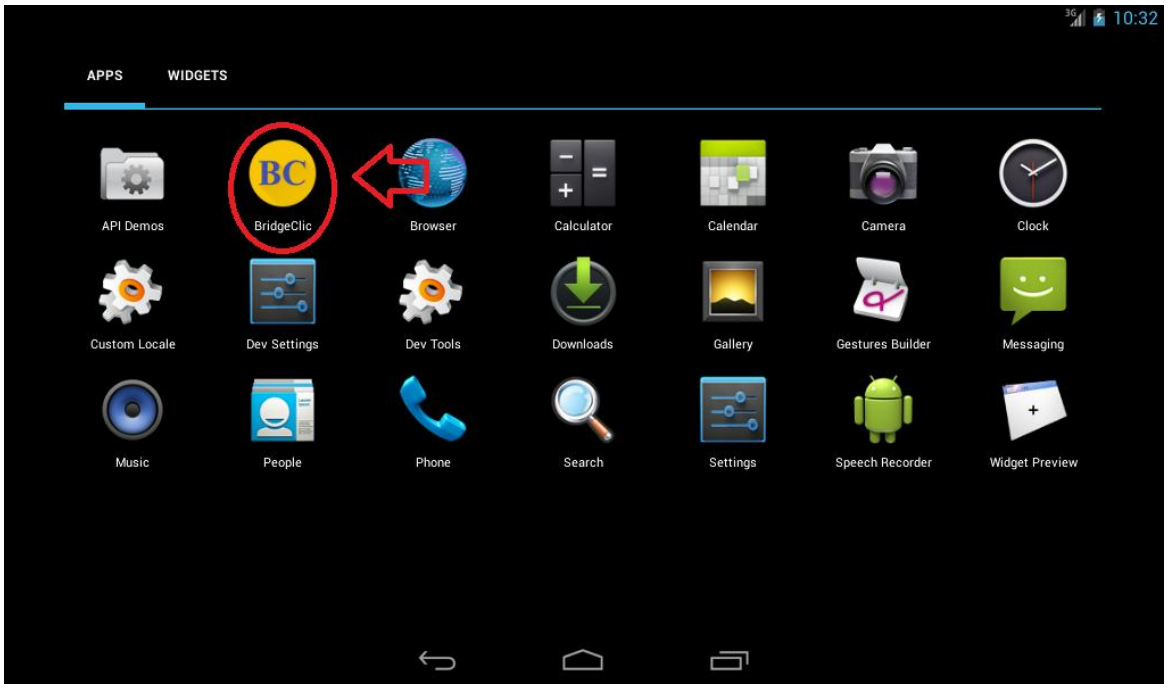

Figura A1.1: Aplicación en el menú del dispositivo.

Una vez iniciada por primera vez mostrará un mensaje de que no existen proyectos en el directorio de la aplicación (ver figura A1.2).

Este será el mensaje mostrado:

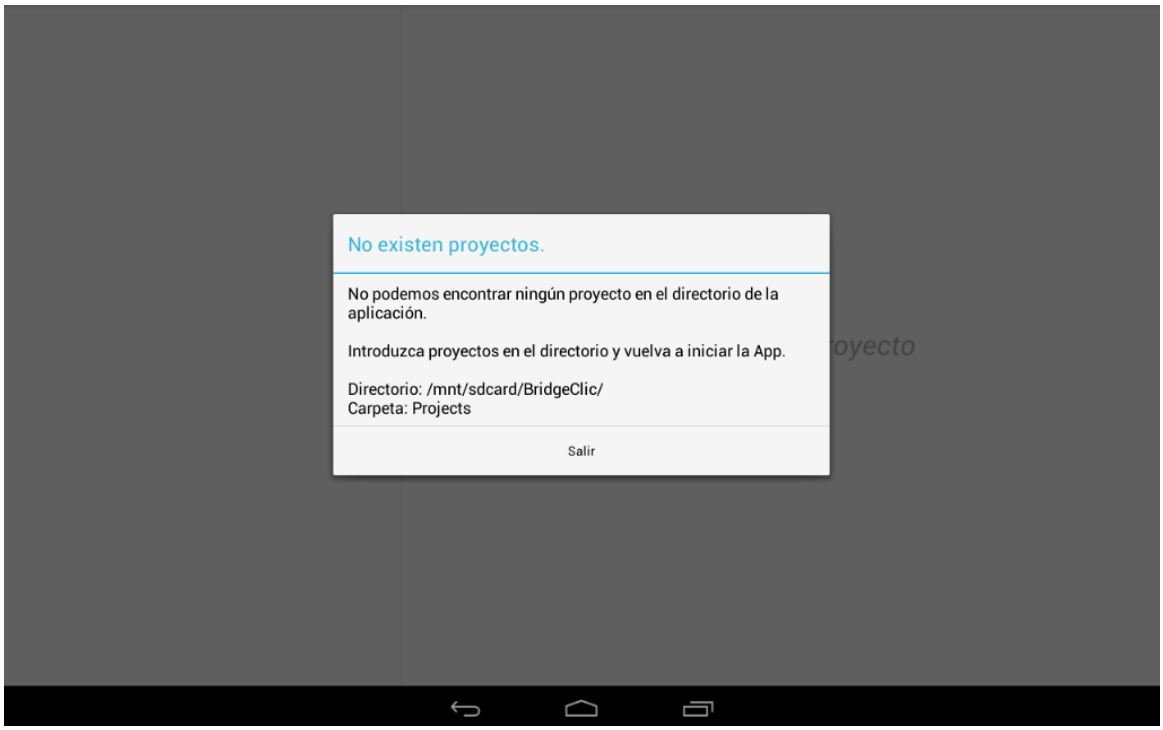

Figura A1.2: Mensaje primer uso aplicación o sin proyectos en el directorio.

Es necesario copiar los proyectos creados con la extensión .Zip por la herramienta de autor, seleccionando incluir las imágenes y demás recursos en el .Zip. El directorio de la aplicación en el que se deben copiar los proyectos se especifica en el mensaje de error.

Uso de la aplicación

Una vez copiados los proyectos en la carpeta especificada se podrá volver a iniciar la aplicación. Esta iniciará una pantalla de carga de proyectos (ver figura A1.3). El tiempo de carga depende de la cantidad de proyectos residentes en el directorio de la aplicación.

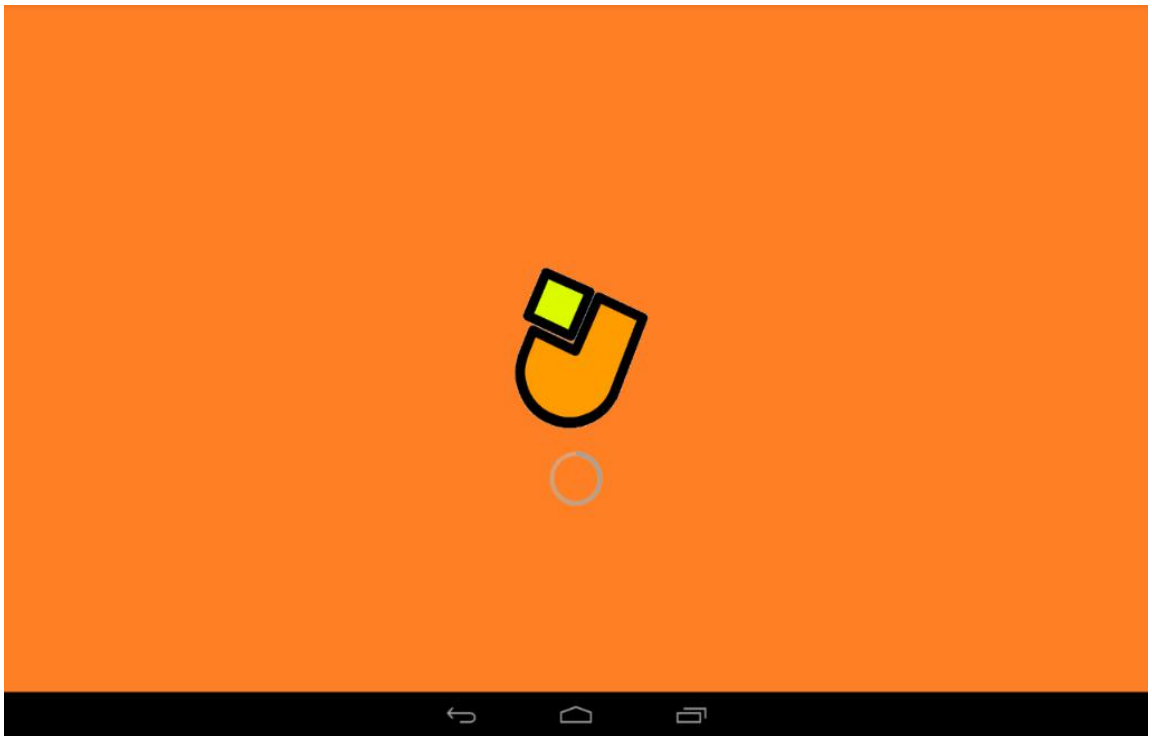

Figura A1.3: Pantalla de carga.

En cuanto la carga finalice se mostrará una pantalla para elegir los proyectos cargados por la aplicación (ver figura A1.4). En el panel de la izquierda se mostrará la lista de proyectos, resaltando el proyecto seleccionado en ese momento y en la parte derecha la descripción del proyecto y un botón para comenzar a jugar. Si existiese alguna actividad no soportada por la aplicación se mostraría una nota en la parte inferior para informar al usuario (ver figura A1.5).

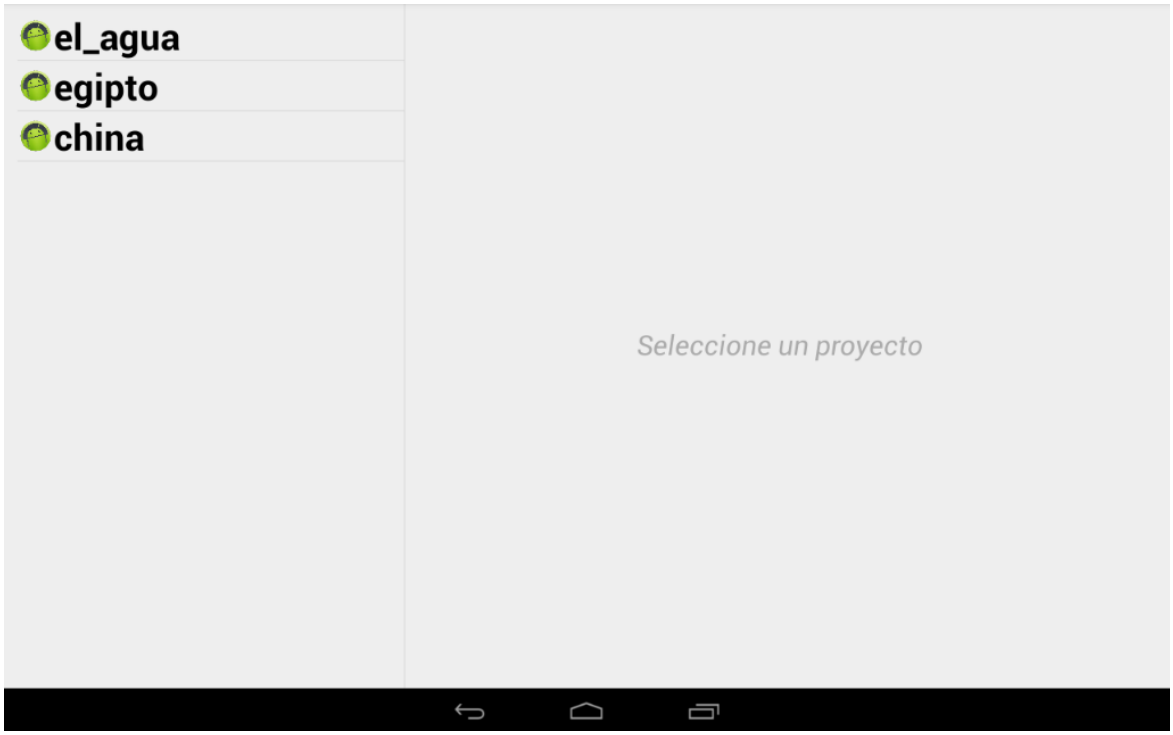

Figura A1.4: Menú de la aplicación sin proyecto seleccionado.

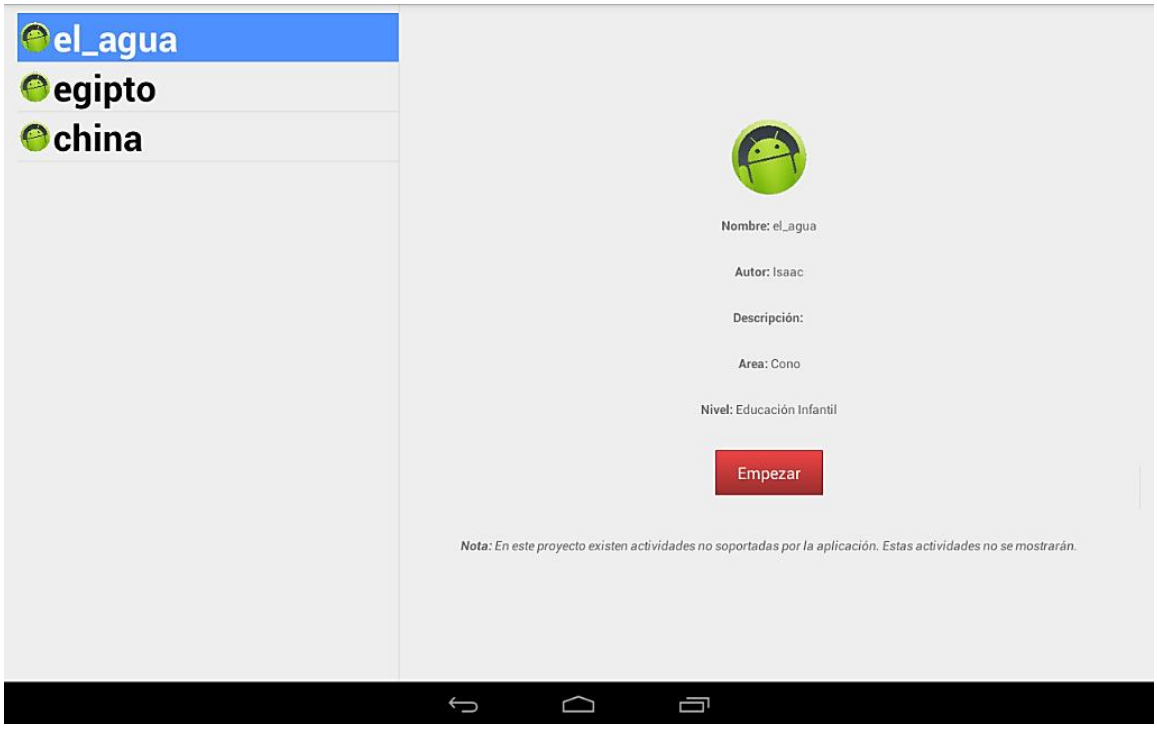

Figura A1.5: Menú de la aplicación con un proyecto seleccionado.

El icono que aparece antes del nombre del proyecto y en la parte superior del panel derecho es el indicado por el proyecto JClic. Si no existe un icono se le asignará el icono por defecto (ver figura A1.5).
Una vez iniciado el proyecto se puede jugar a cada una de las actividades del proyecto siempre que estas estén soportadas. Estos son un ejemplo de dichas actividades:

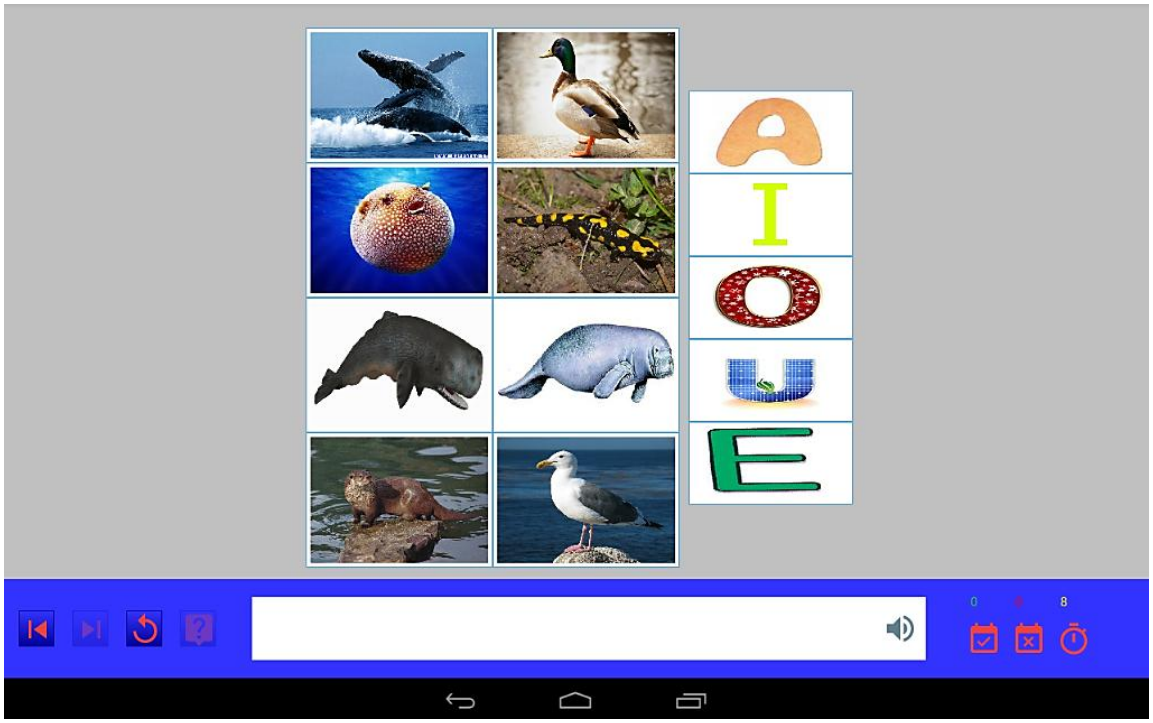

Figura A1.6: Ejemplo asociación compleja.

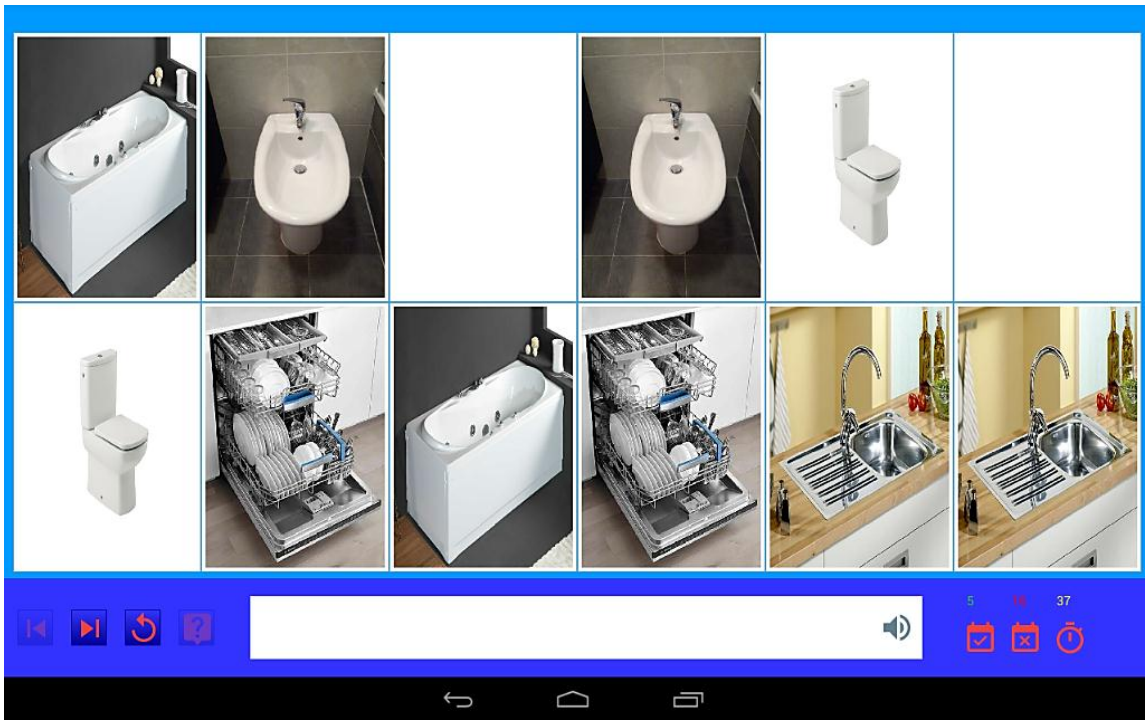

Figura A1.7: Ejemplo juego memoria.

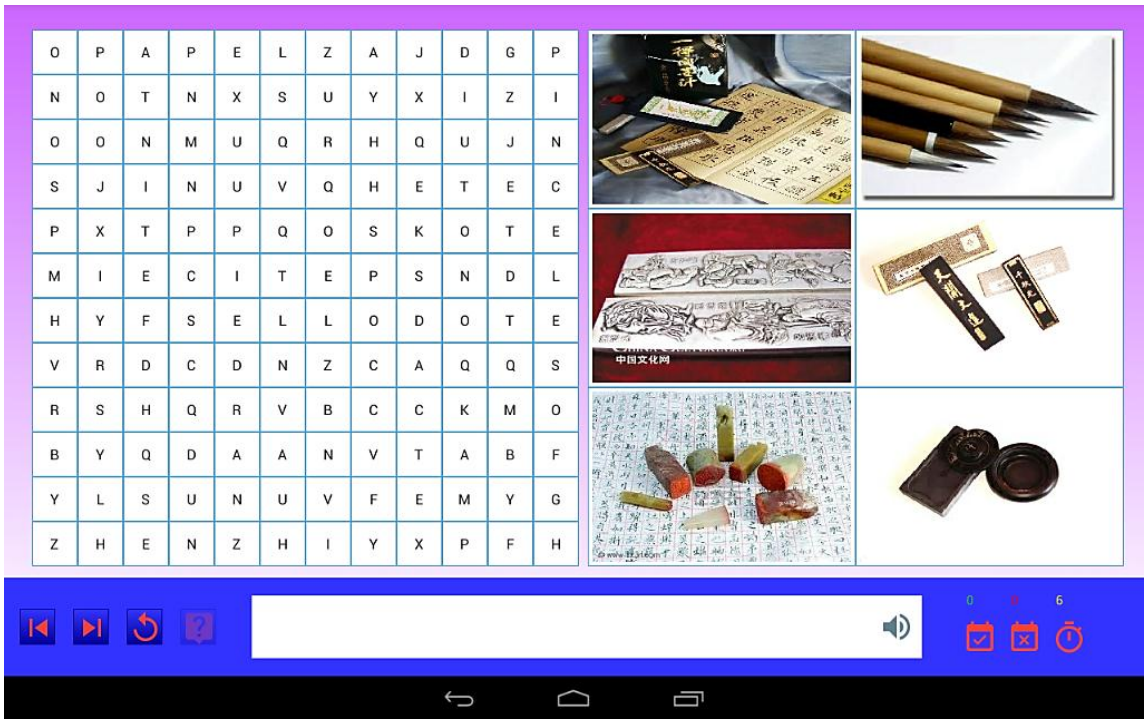

Figura A1.8: Ejemplo sopa de letras.

El panel inferior dispone de 4 botones:

- Botón para retroceder a la actividad anterior.
- Botón para avanzar a la actividad siguiente.
- Botón para repetir la actividad.
- Botón para mostrar la ayuda de la actividad.

Estos botones estarán habilitados o deshabilitados dependiendo de las posibilidades que existan en ese momento. El display de la parte inferior derecha muestra los aciertos, fallos y tiempo de la actividad (de izquierda a derecha). En la parte central del panel inferior se encuentra la descripción de la actividad. Si se quiere escuchar la descripción solo es necesario pulsar sobre la misma.

Mensajes de la aplicación

La aplicación mostrará diferentes mensajes dependiendo de los eventos lanzados. El mensaje que se mostrará depende de lo que autor tenga especificado en el proyecto. Estos son unos cuantos ejemplos de mensajes que nos podemos encontrar en la aplicación.

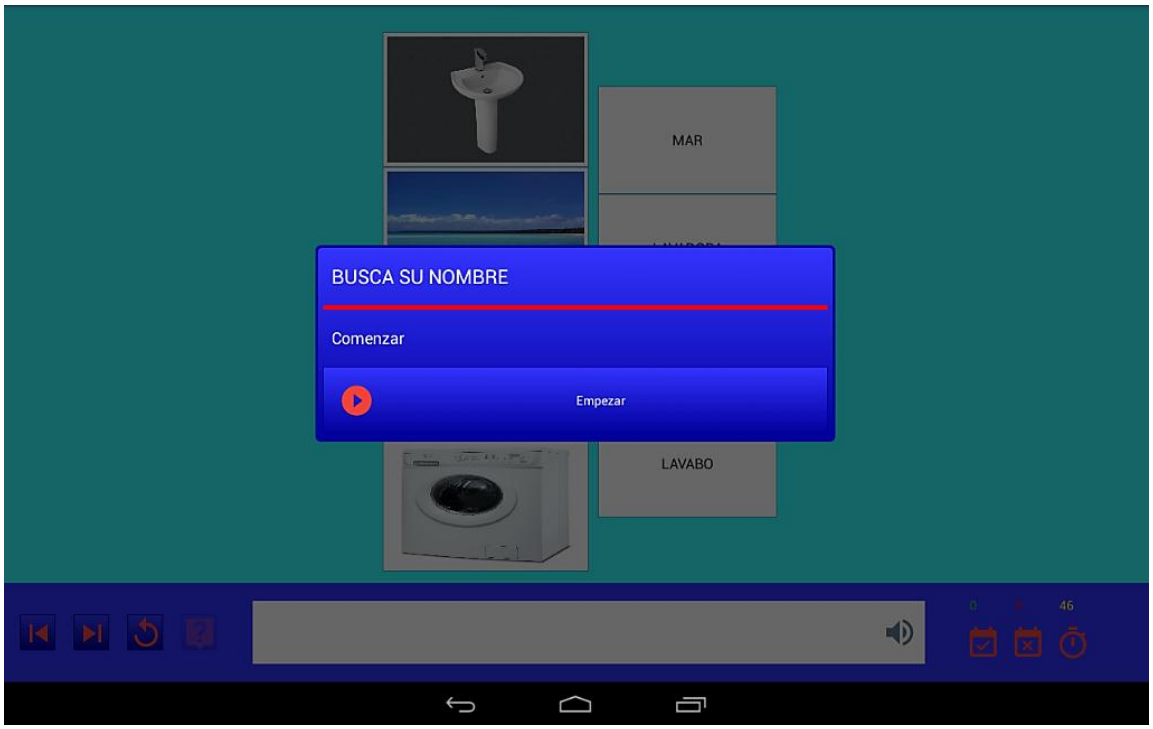

Figura A1.9: Mensaje de inicio de actividad.

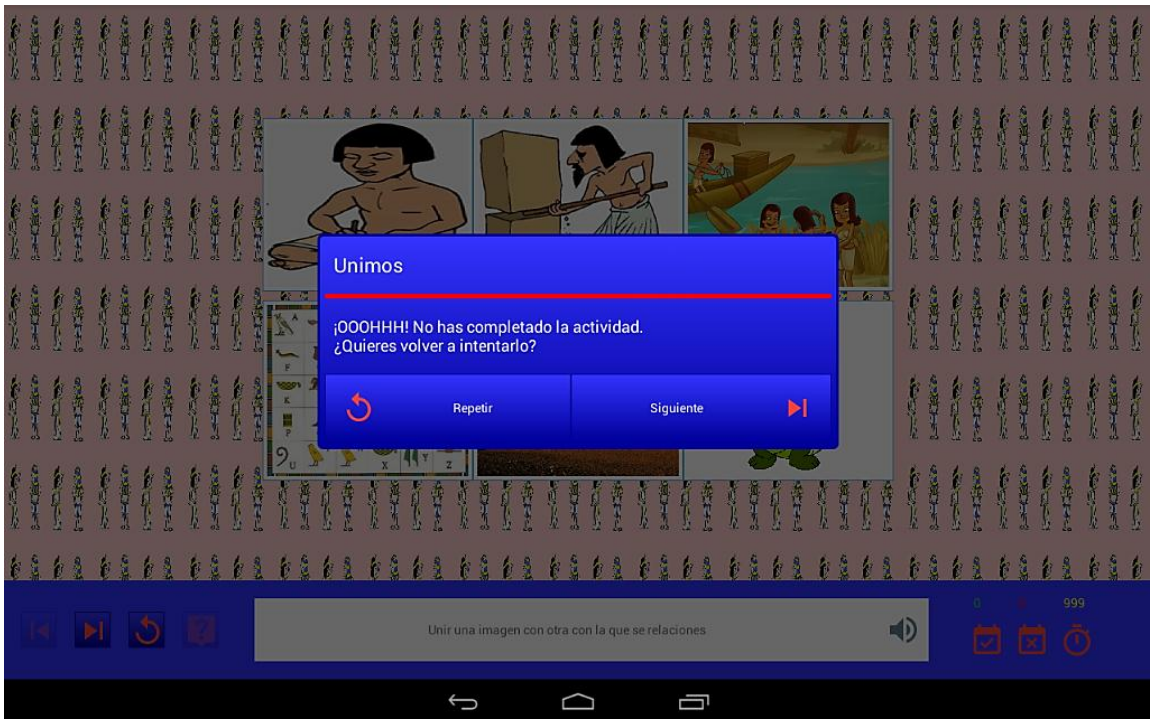

Figura A1.10: Mensaje fin actividad.

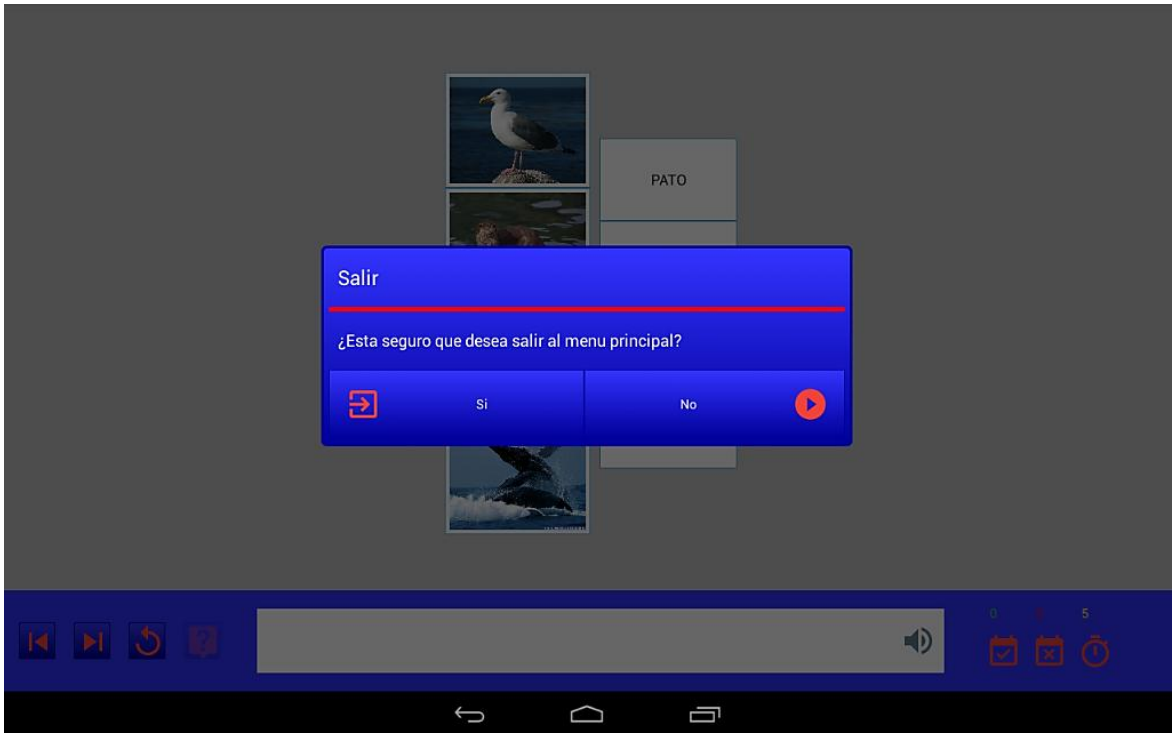

Figura A1.11: Mensaje salir proyecto.

Estas serían las pautas para ejecutar la aplicación. Las imágenes mostradas en el manual solo son un primer vistazo a la App, dependiendo de los proyectos creados se podrán ver diferentes resultados.

Anexo 2: Ejemplo XML JClic

```
<?xml version="1.0" encoding="UTF-8"?>
<JClicProject name="proyectotfg" version="0.1.3">
 <settings>
  <title>ProyectoTFG</title>
  <revision description="created" date="11/4/14" />
  <author name="Miguel" mail="correo" url="url-texto" rol="Director"
organization="organizacion-TFG">
   <comments>
    <p>text area comentarios</p>
   </comments>
  </author>
  <organization name="Universidad autonoma" mail="correo" url="url" address="dir"
pc="22222" city="madrid" country="españa" state="madrid">
   <comments>
    <p>comentario</p>
   </comments>
  </organization>
  <language>español (es)</language>
  <description>
   <p>Descripcion del proyecto</p>
  </description>
  <descriptors area="infantil" level="3 tipos de tareas">tareas para TFG</descriptors>
  <skin file="@orange.xml" />
 </settings>
 <sequence>
  <item id="Actividad 1" name="Asociacion compleja" delay="1">
   <description>
    <p>Asociacion compleja</p>
   </description>
  </item>
 </sequence>
 <activities>
  <activity class="@associations.ComplexAssociation" name="Asociacion compleja"
code="1">
   <description>
    <p>Actividad de asociacion compleja</p>
   </description>
   <messages>
    <cell border="true" type="initial">
     <style />
     <p>Asociacion compleja</p>
    </cell>
    <cell type="final">
     <style />
     <p>Mensaje final asociacion compleja</p>
    </cell>
    <cell type="finalError">
     <style />
```

```
    <p>mensaje de error asociacion compleja</p>
   </cell>
  </messages>
  <settings margin="5" maxTime="10" countDownTime="false" report="true"
reportActions="false">
   <helpWindow showSolution="false" />
   <container bgColor="0x3333FF">
    <counters time="true" actions="true" score="true" />
   </container>
   <window bgColor="0xFFFF00" border="true" />
   <skin file="@orange.xml" />
  </settings>
  <cells rows="2" cols="3" cellWidth="50.0" cellHeight="28.0" border="true"
id="primary">
   <style />
   <shaper class="@Rectangular" cols="3" rows="2" />
   <cell id="0">
    <p>a</p>
   </cell>
   <cell id="1">
    <p>b</p>
   </cell>
   <cell id="2">
    <p>c</p>
   </cell>
   <cell id="3">
    <p>d</p>
   </cell>
   <cell id="4">
    <p>e</p>
   </cell>
   <cell id="5">
    <p>f</p>
   </cell>
  </cells>
  <cells rows="2" cols="3" cellWidth="50.0" cellHeight="30.0" border="true"
id="secondary">
   <style />
   <shaper class="@Rectangular" cols="3" rows="2" />
   <cell>
    <p>a</p>
   </cell>
   <cell>
    <p>b</p>
   </cell>
   <cell>
    <p>c</p>
   </cell>
   <cell>
    <p>d</p>
```

```
    </cell>
    <cell>
     <p>e</p>
    </cell>
    <cell>
     <p>f</p>
    </cell>
   </cells>
   <scramble times="31" primary="true" secondary="true" />
   <layout position="AB" />
  </activity>
 </activities>
 <mediaBag>
  <media name="gato.png" file="gato.png" />
  <media name="perro.png" file="perro.png" />
 </mediaBag>
</JClicProject>
```